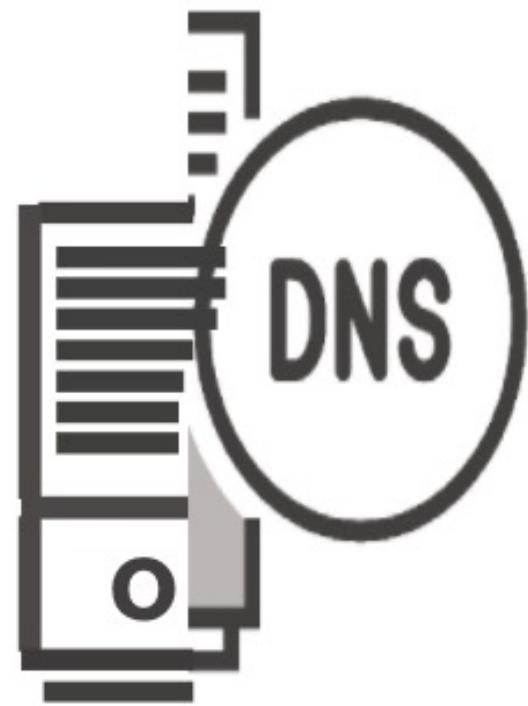


Despliegue de Aplicaciones/Sistemas Distribuidos

**Fundamentos y Tecnologías
Parte2**

Desplegando Aplicaciones

Desplegando Aplicaciones Tradicionalmente (1)



X Cores
XX GB Ram
XXXX GB HDD



X Cores
XX GB Ram
XXXX GB HDD



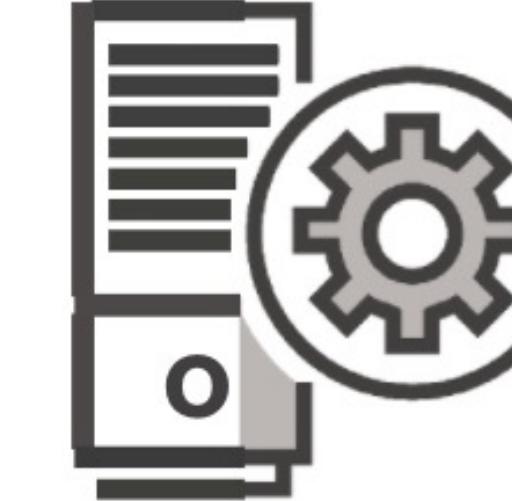
X Cores
XX GB Ram
XXXX GB HDD



X Cores
XX GB Ram
XXXX GB HDD



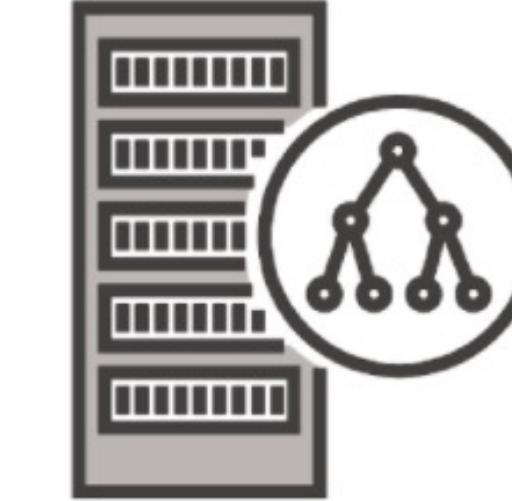
X Cores
XX GB Ram
XXXX GB HDD



X Cores
XX GB Ram
XXXX GB HDD



X Cores
XX GB Ram
XXXX GB HDD



X Cores
XX GB Ram
XXXX GB HDD

Desplegando Aplicaciones Tradicionalmente (2)



12% Utilización

X Cores Ghz
XX GB Ram
XXXX GB HDD



30% Utilización

X Cores
XX GB Ram
XXXX GB HDD



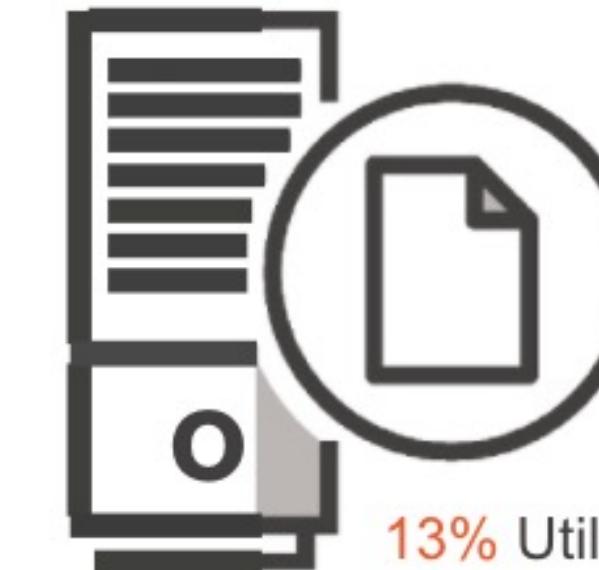
25% Utilización

X Cores Ghz
XX GB Ram
XXXX GB HDD



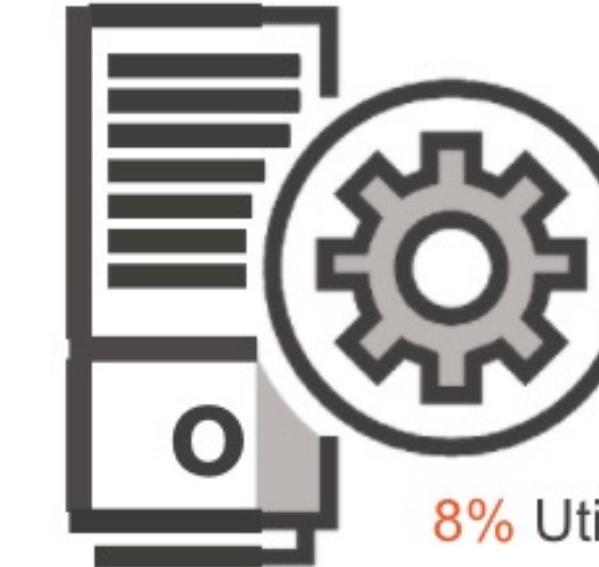
30% Utilización

X Cores
XX GB Ram
XXXX GB HDD



13% Utilización

X Cores
XX GB Ram
XXXX GB HDD



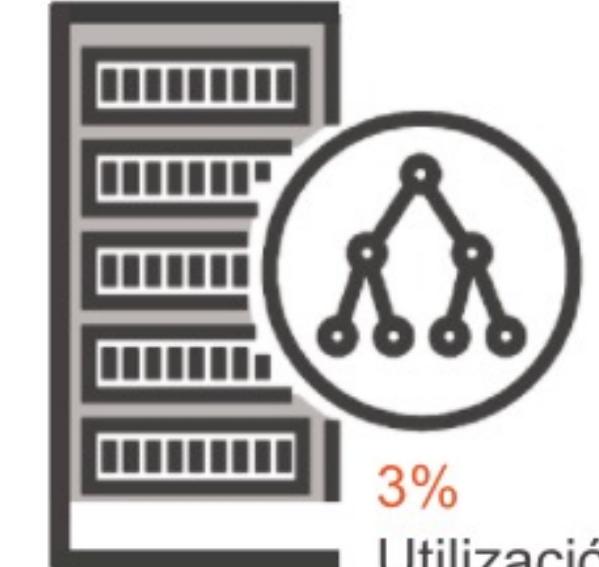
8% Utilización

X Cores
XX GB Ram
XXXX GB HDD



5% Utilización

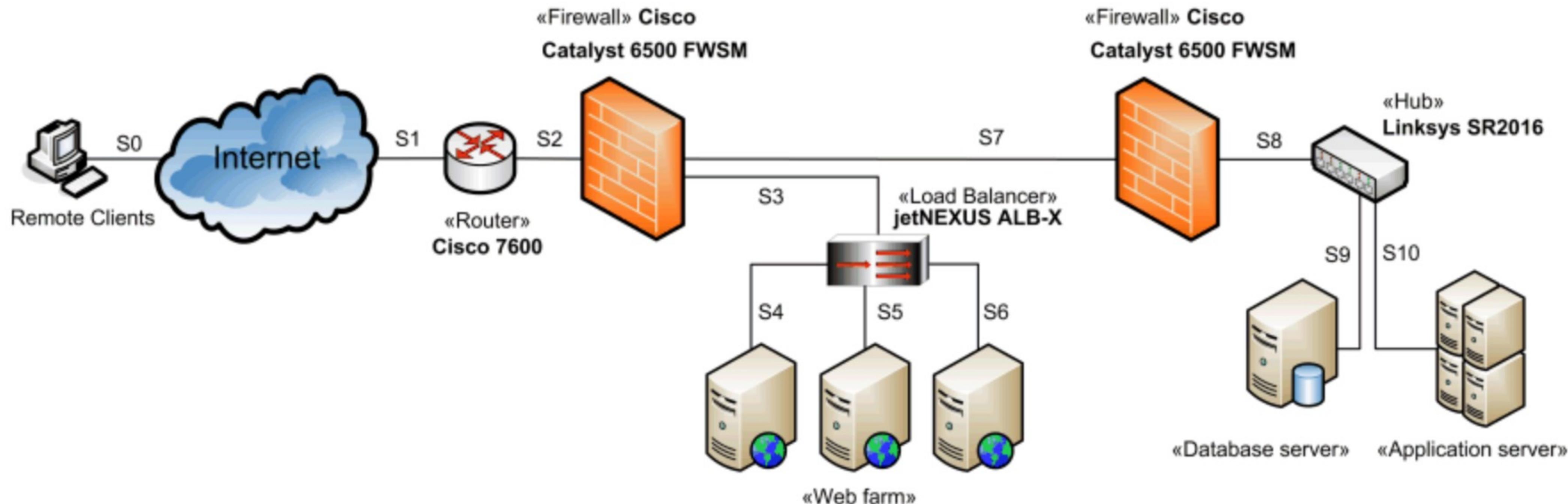
X Cores
XX GB Ram
XXXX GB HDD



3% Utilización

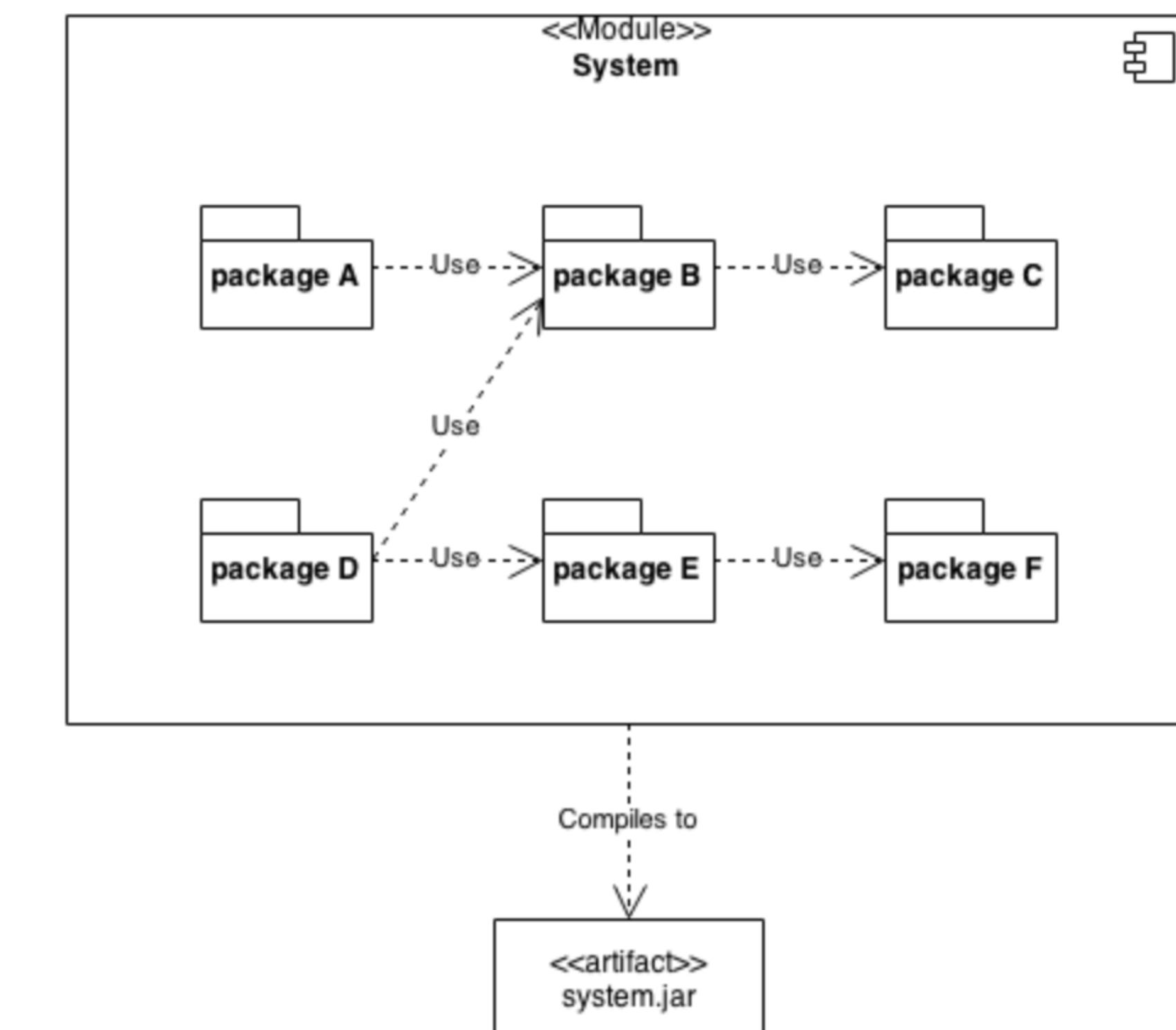
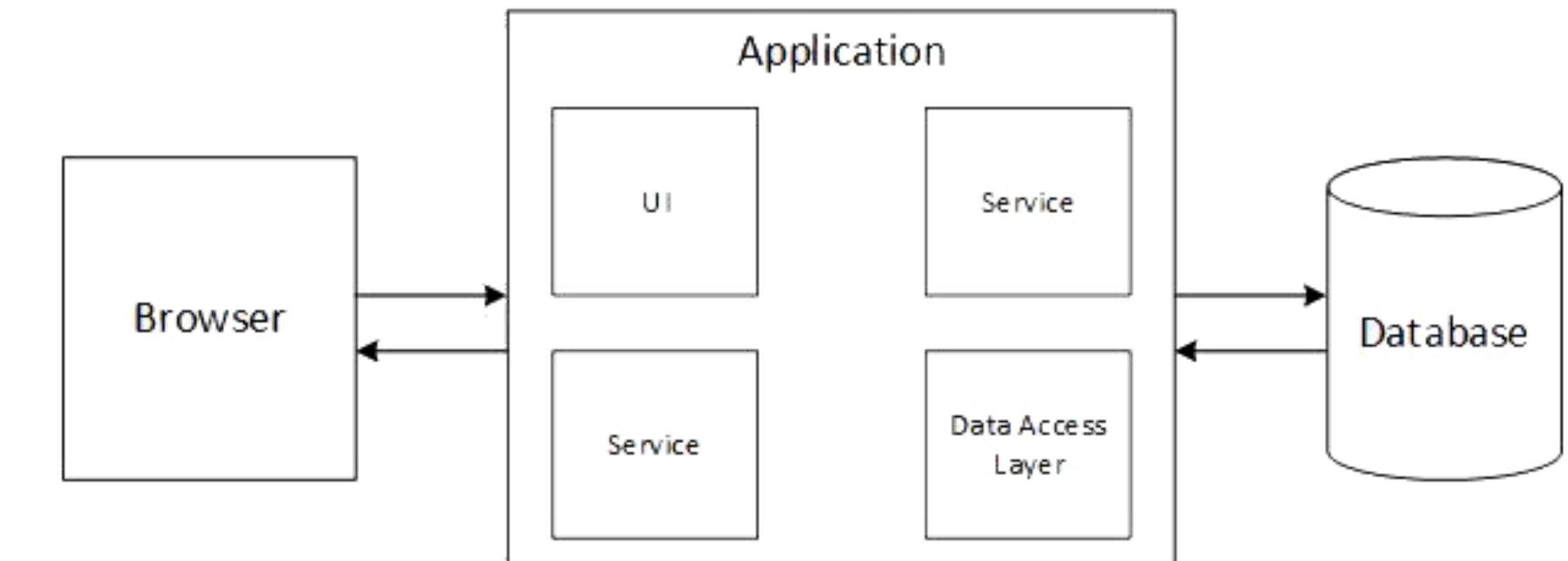
X Cores
XX GB Ram
XXXX GB HDD .

Topología para el Despliegue Tradicional de Aplicaciones On-Premise



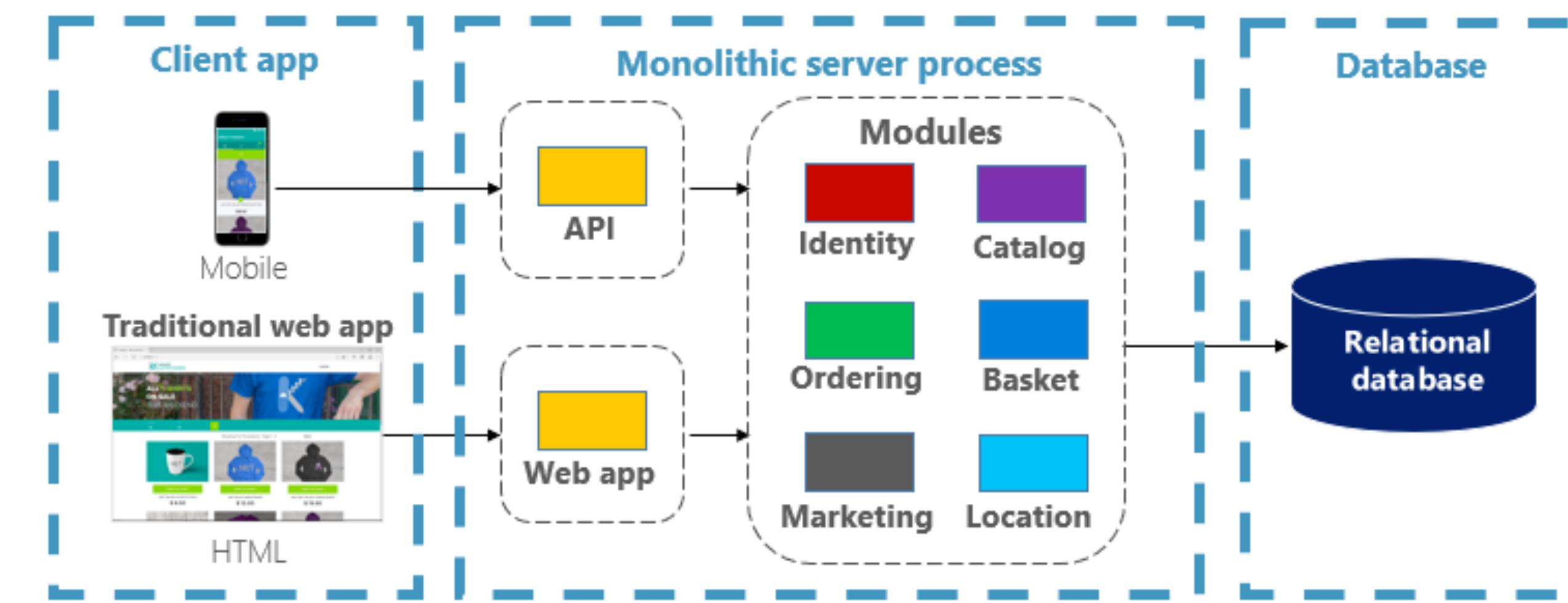
Desplegando Aplicaciones Monolíticas

- En este contexto, cuando se hace referencia a una aplicación monolítica, es a una sola unidad de despliegue.
- Esto quiere decir que toda la funcionalidad de un sistema, debe ser desplegada de manera conjunta.
- Todo el código es empaquetado y ejecutado como un único proceso.

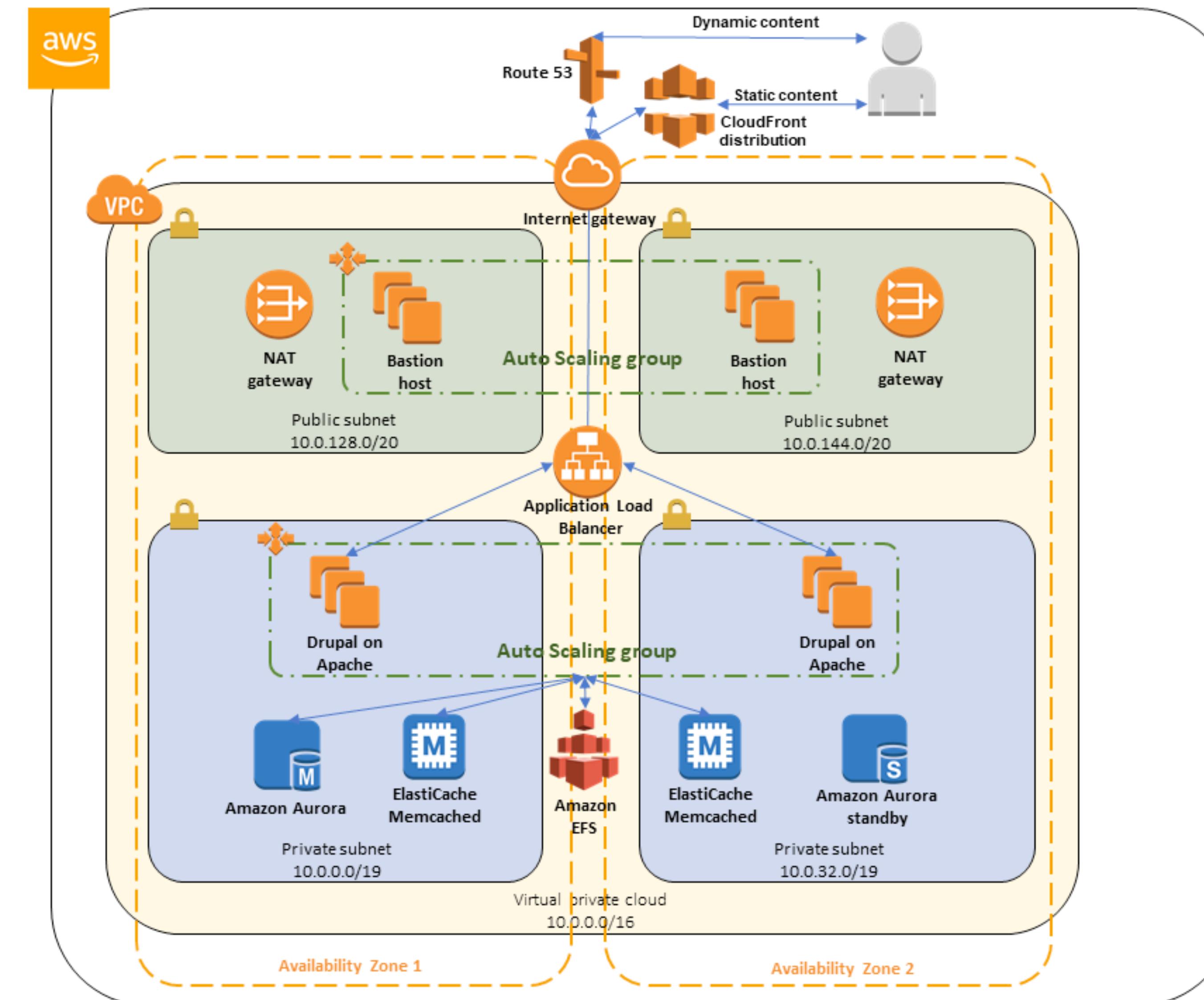


Desplegando Aplicaciones Monolíticas

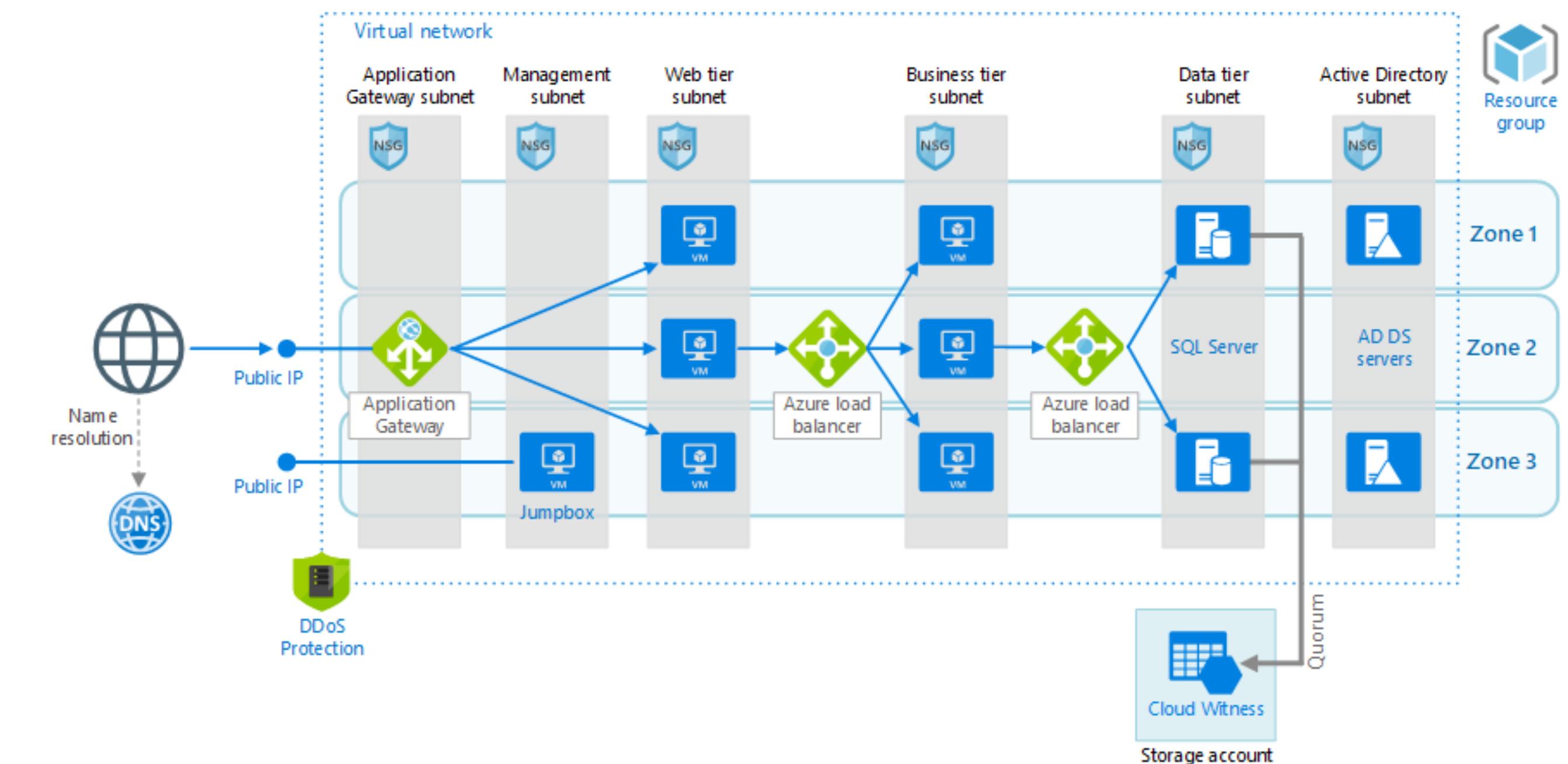
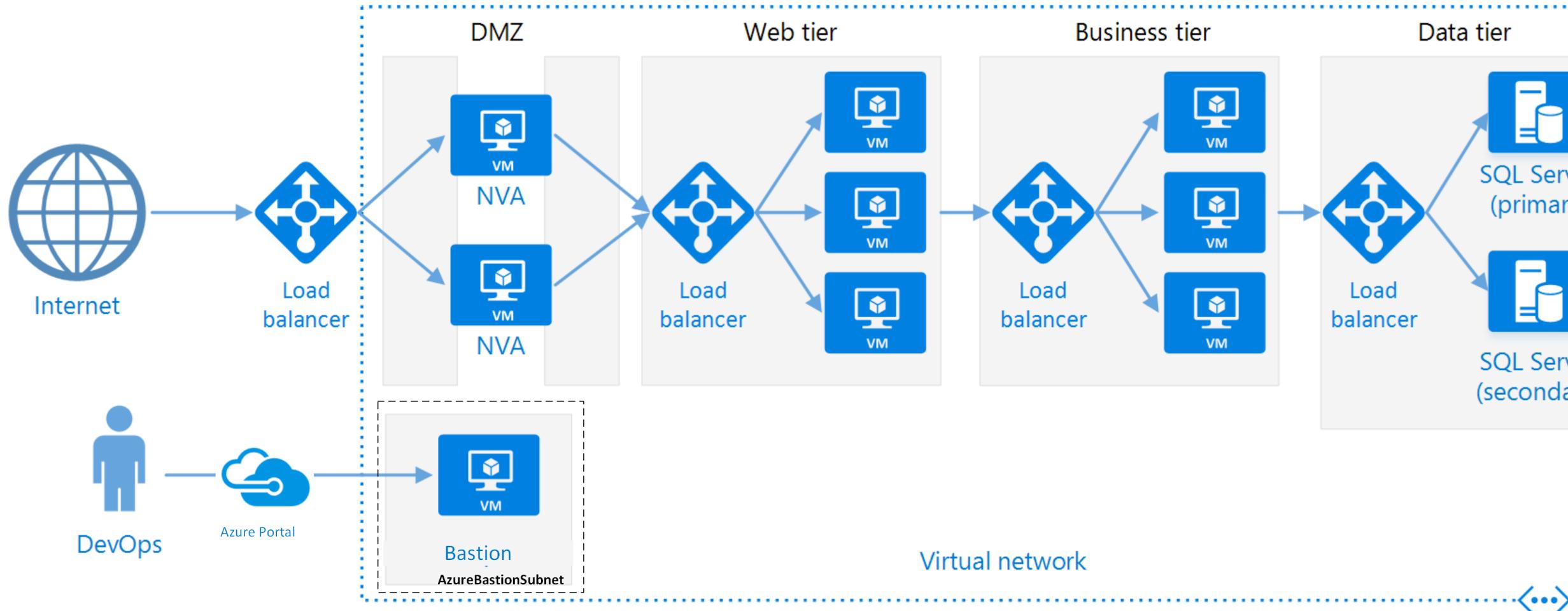
- La escalabilidad de este tipo de aplicaciones, implica un hardware muy potente.
 - Si cualquier parte de la aplicación requiere escalar, se hace necesario escalar toda la aplicación en otra máquina. No se puede escalar un componente de forma individual.
 - Si se hizo una actualización a algún módulo, se hace necesario reiniciar todo el sistema.
 - Este tipo de aplicaciones normalmente son construidas en un solo lenguaje de programación



Ejemplo de Arquitectura de Referencia para Despliegue de una Aplicación Monolítica usando un CMS



Desplegando una N-Tier Web Application



Tecnologías para el Despliegue de Aplicaciones

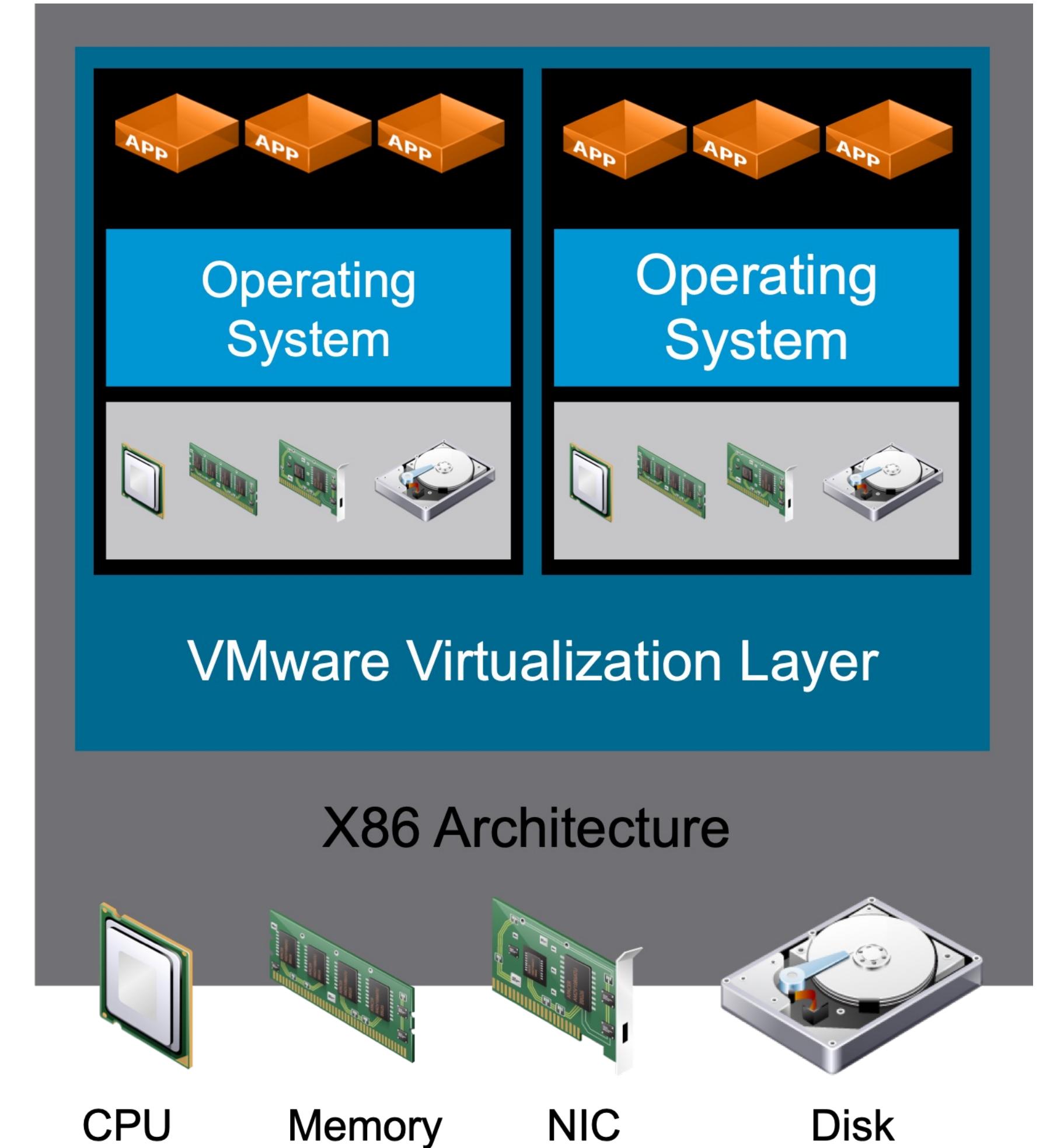
Tecnologías Empleadas para el Despliegue de Aplicaciones

- Virtualización.
- Contenedores.
 - Kubernetes.
- Serveless.

Virtualización

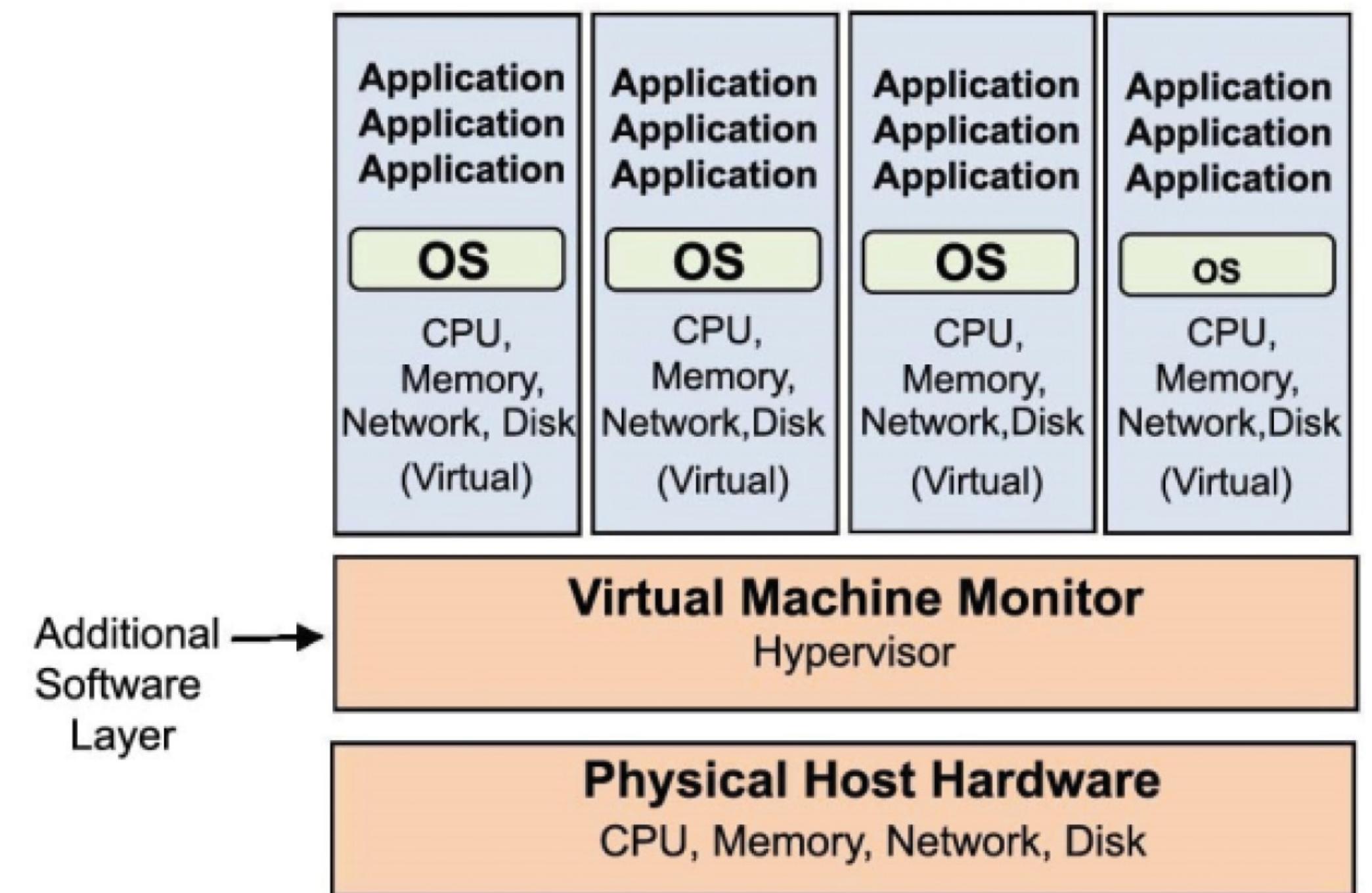
Virtualización

- Usualmente los PCs corren solo un sistema operativo.
- Cambiar entre sistemas operativos o instancias usualmente requiere una máquina separada.
- De esta forma existe una forma de correr dos o mas sistemas operativos de manera concurrente sobre un mismo HW/máquina:
 - Virtualización.



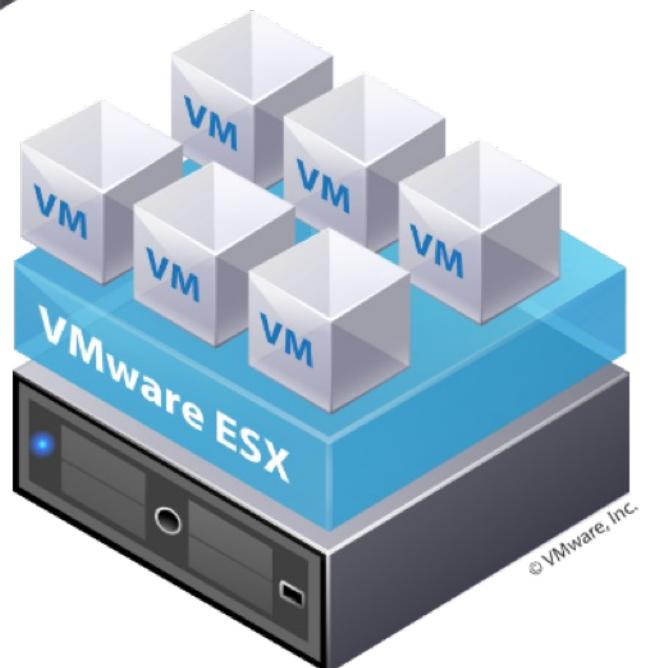
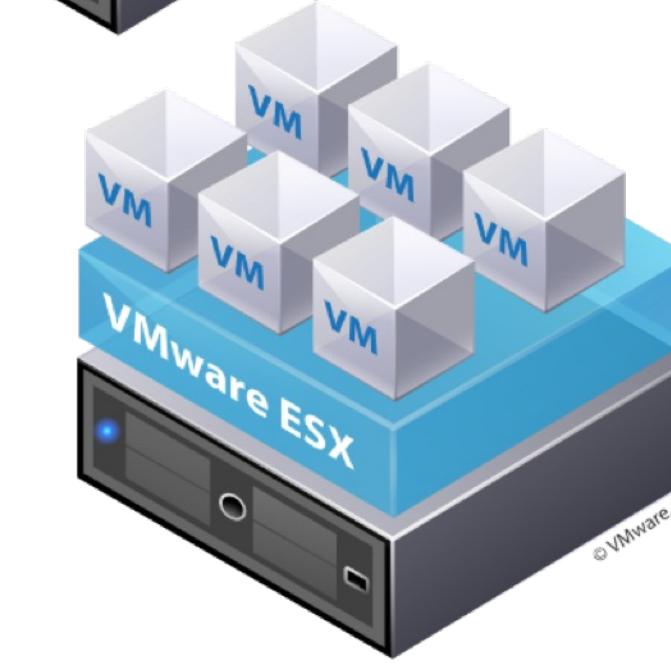
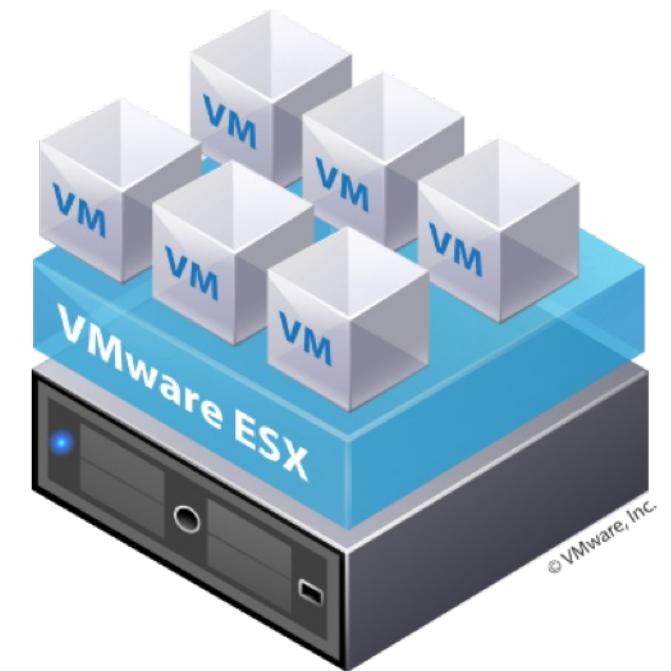
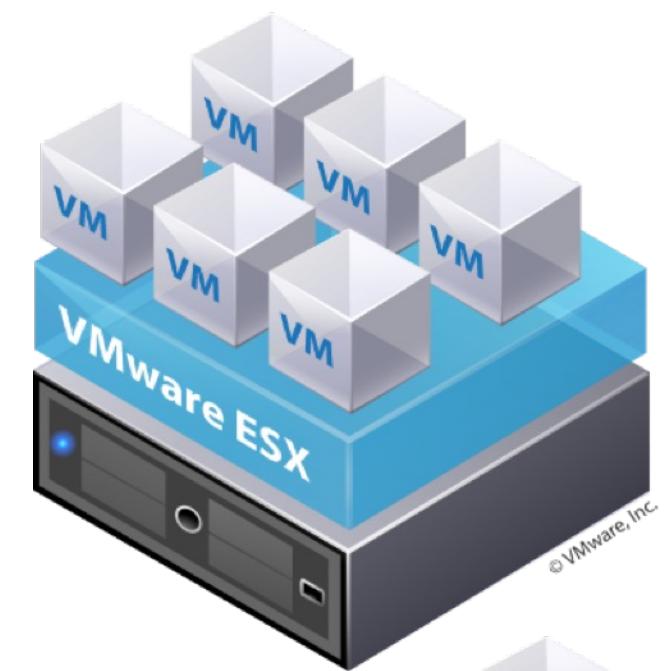
Virtualización

- La virtualización se refiere a la creación de una versión virtual de un dispositivo o recurso tal como un server, recurso de almacenamiento o red donde el framework divide el recurso en uno o mas entornos de ejecución.
- En otras palabras es un técnica de software que habilita esta técnica es denominado hypervisor.
 - Se convierte en una interfaz entre este nuevo sistema operativo y el HW.
- De esta forma la máquina virtual (VM) es esencialmente construido como un paquete software que puede ser cargado en un host para ejecutar ciertas aplicaciones.
- Múltiples VMs pueden co-existir en el mismo host siempre y cuando se tenga la memoria suficiente para soportar las máquinas virtuales.



Retos de la Virtualización

- Existen algunos retos de la virtualización:
 - Al comienzo, muchas empresas de SW se mostraron reacias a certificar sus soluciones en ambientes de virtualización.
 - Una falla física puede afectar múltiples máquinas virtuales.
 - La gestión de almacenamiento es puede ser desafiante.
 - La alta consolidación de máquinas virtuales en un solo servidor físico puede derivar en servidores mas complejos.



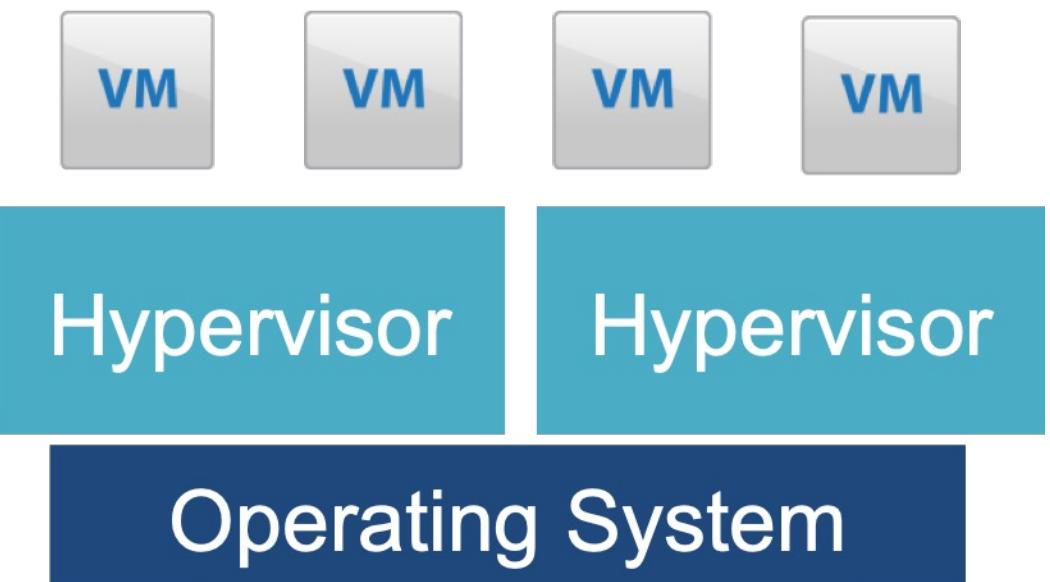
Tipos de Virtualización

- El software que implementa y gestiona los aspectos de virtualización se denominan hypervisor.
- El hypervisor crea y gestiona las máquinas virtuales.
- Facilita el acceso compartido a los recursos físicos de la máquina.

Type 1 Hypervisor

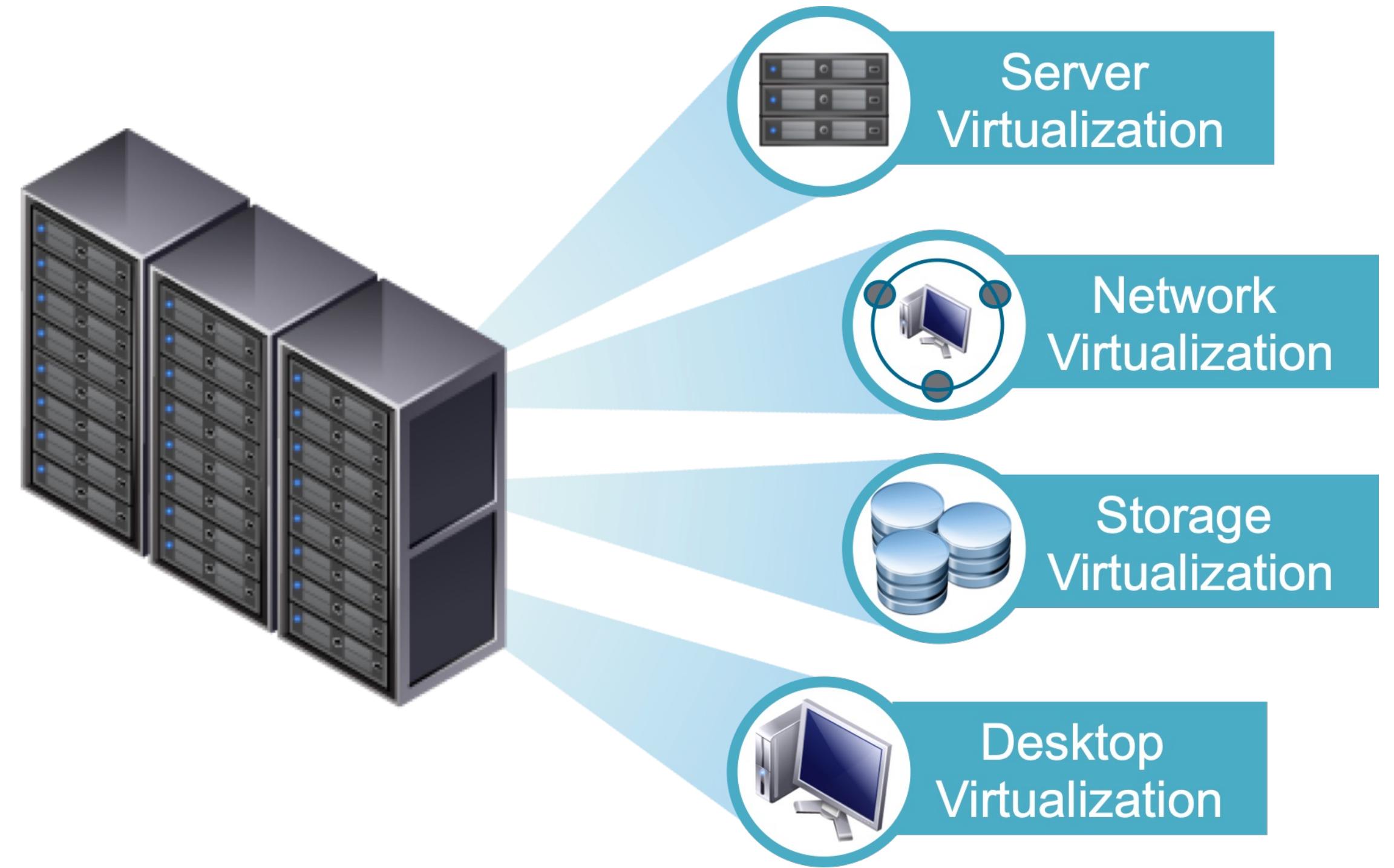


Type 2 Hypervisor



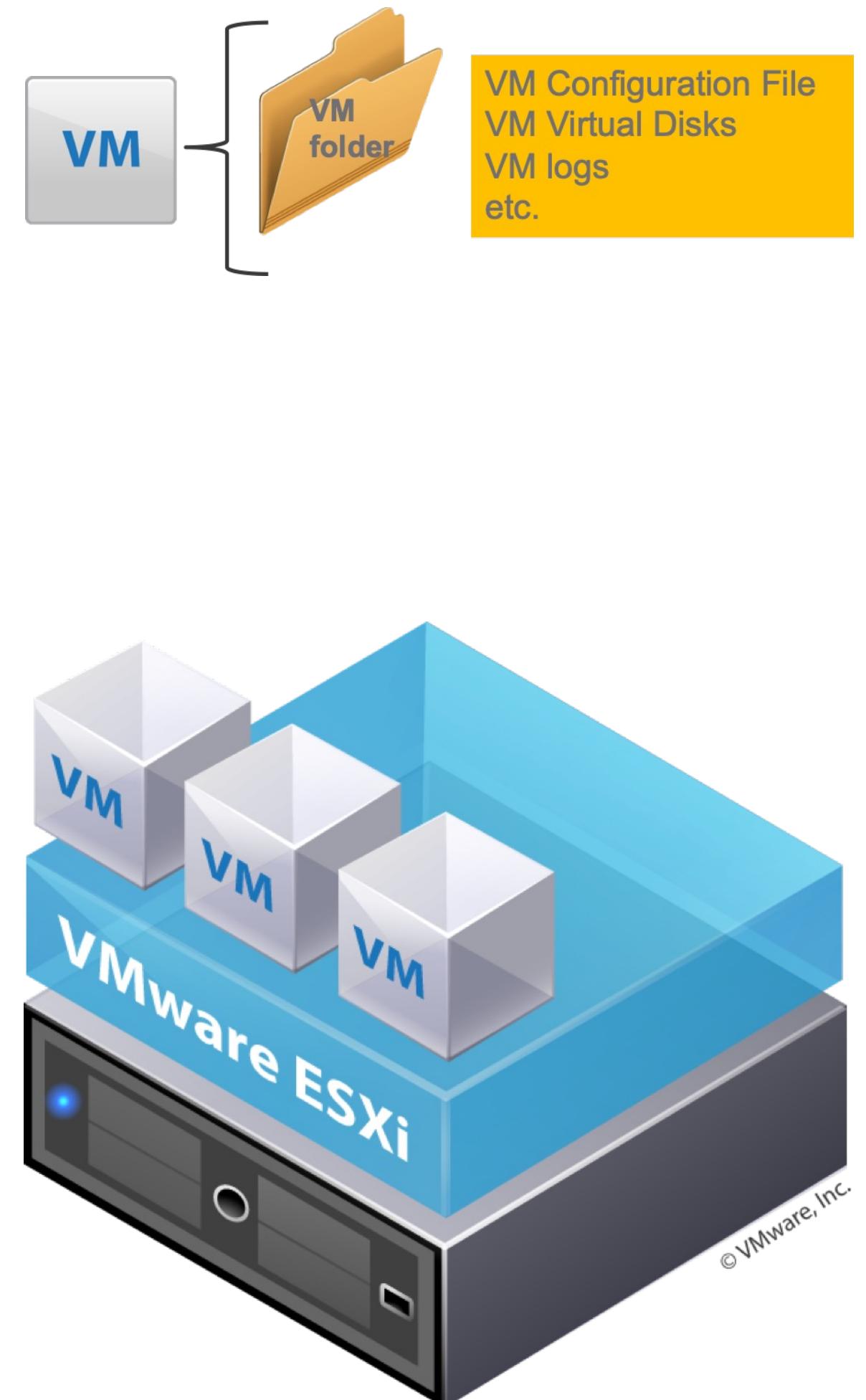
Tecnologías de Virtualización

- La virtualización requiere la habilidad efectiva para gestionar los recursos a nivel de cómputo, almacenamiento y las redes.
- Cada vez mas los diseñadores de soluciones tienden a apalancar sus soluciones con infraestructura compartida



Máquinas Virtuales

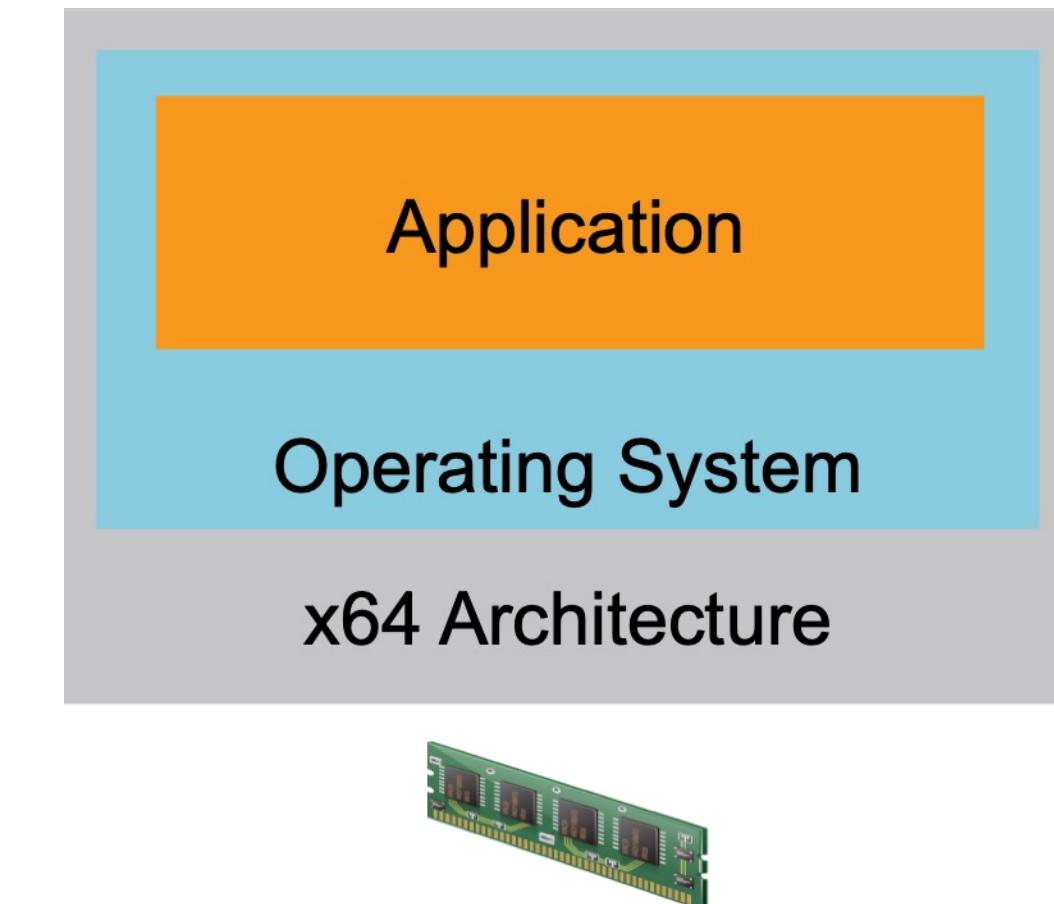
- Las máquinas virtuales son el componente fundamental de la virtualización.
- Puede ser entendidos como los contenedores para los sistemas operativos tradicionales y las aplicaciones que corren sobre el hipervisor.
- Al interior de una máquina virtual, todo es muy parecido a como en un servidor físico.
- Una máquina virtual es un conjunto de archivos que describen el servidor físico.
 - Las máquinas virtuales tienen acceso a varios recursos de HW.
 - Desde el punto de vista de la máquina virtual, ella no sabe que esos recursos no existen.
 - Las máquinas virtuales acceden a dispositivos virtuales. Se construyen representaciones virtuales de éstos.



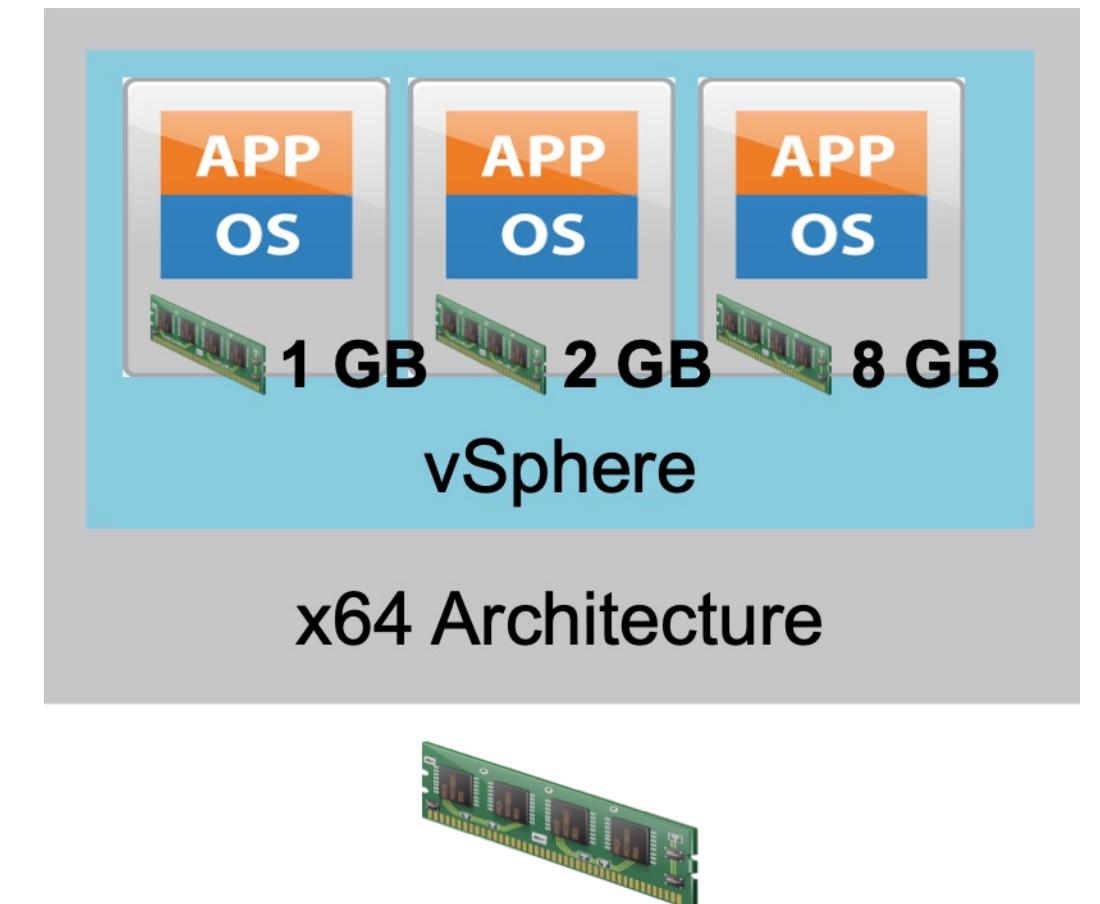
CPU y Memoria

- Las máquinas virtuales están configuradas para correr con uno o mas CPUs.
- En el caso mas simple una máquina virtual, tendrá asignada un CPU.
- Lo que se asigna, desde la perspectiva del hosts, es la capacidad para que la máquina virtual solicite tiempos de CPU.
- En este caso es el hipervisor realiza la solicitud de los recursos en nombre de la máquina virtual.

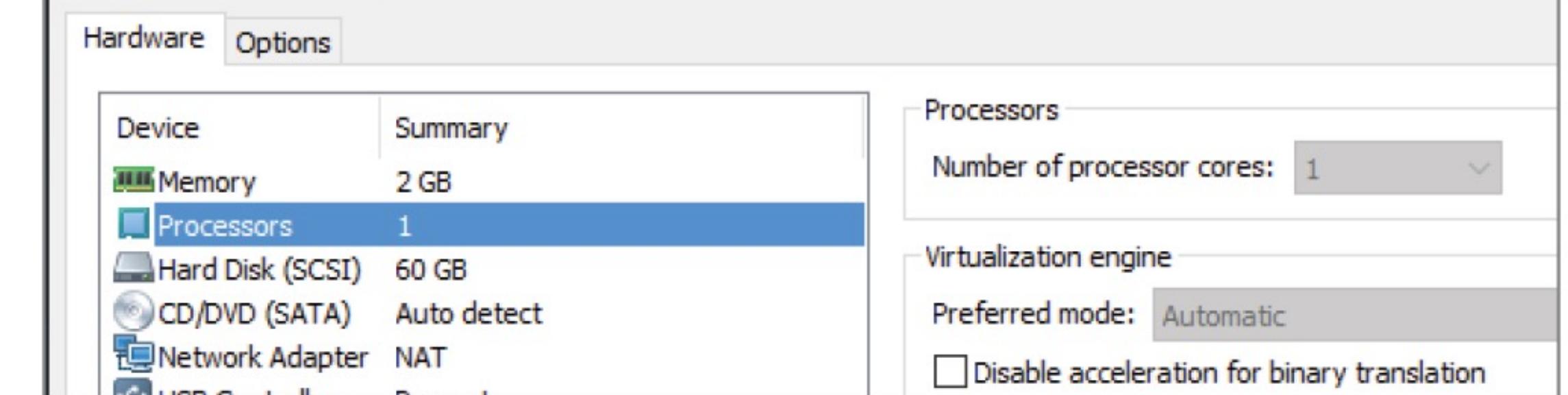
Physical Architecture



Virtual Architecture

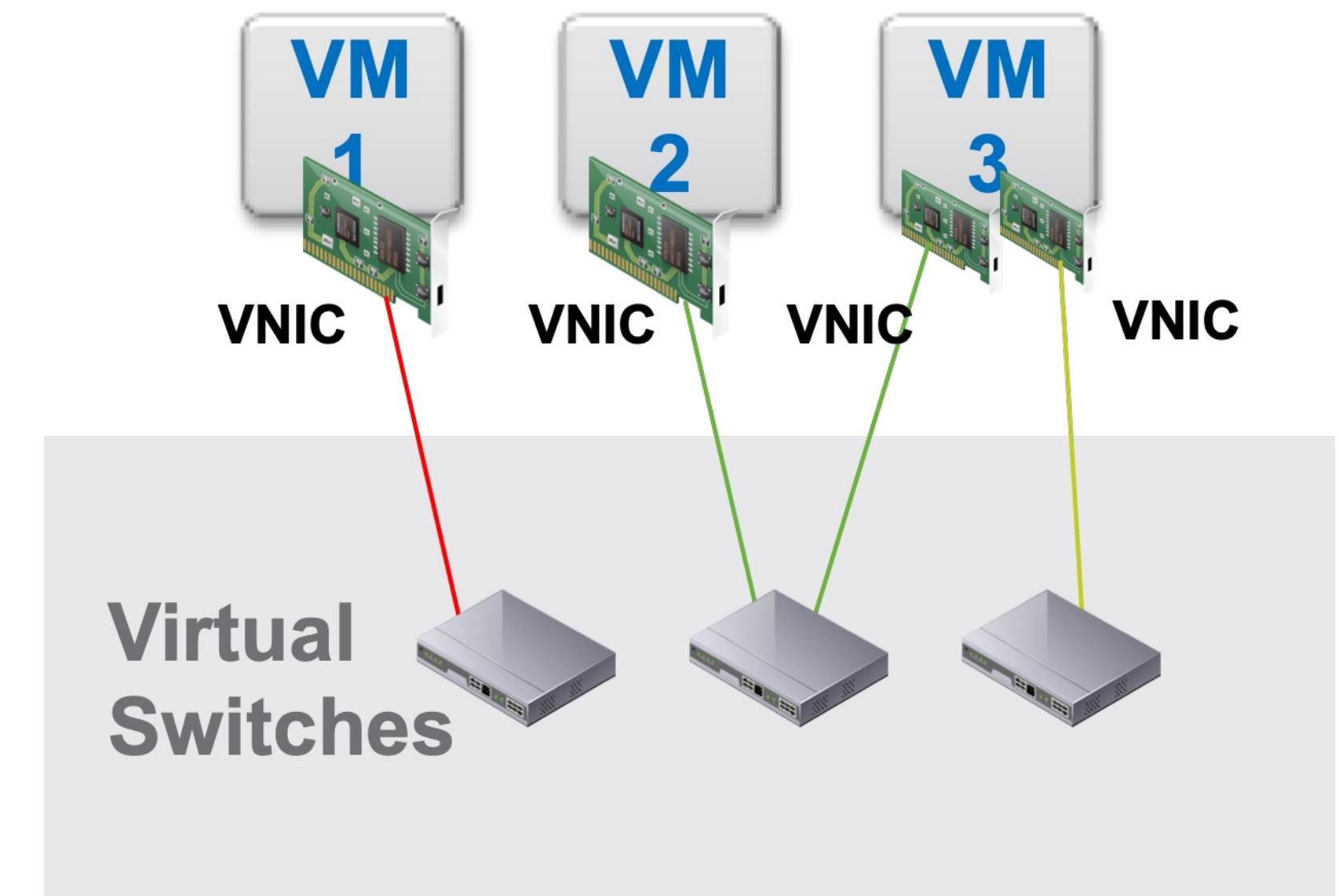
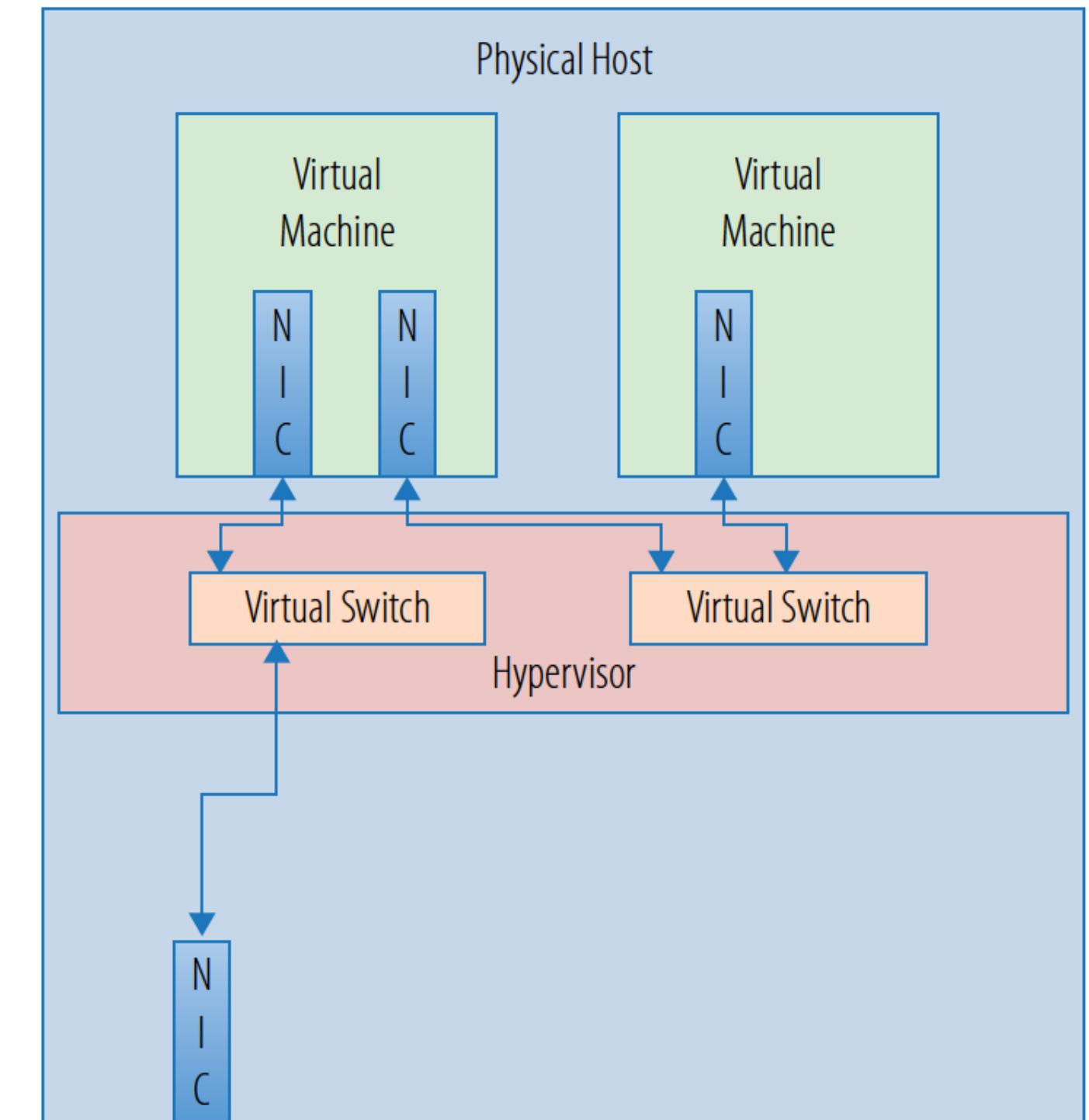


Virtual Machine Settings



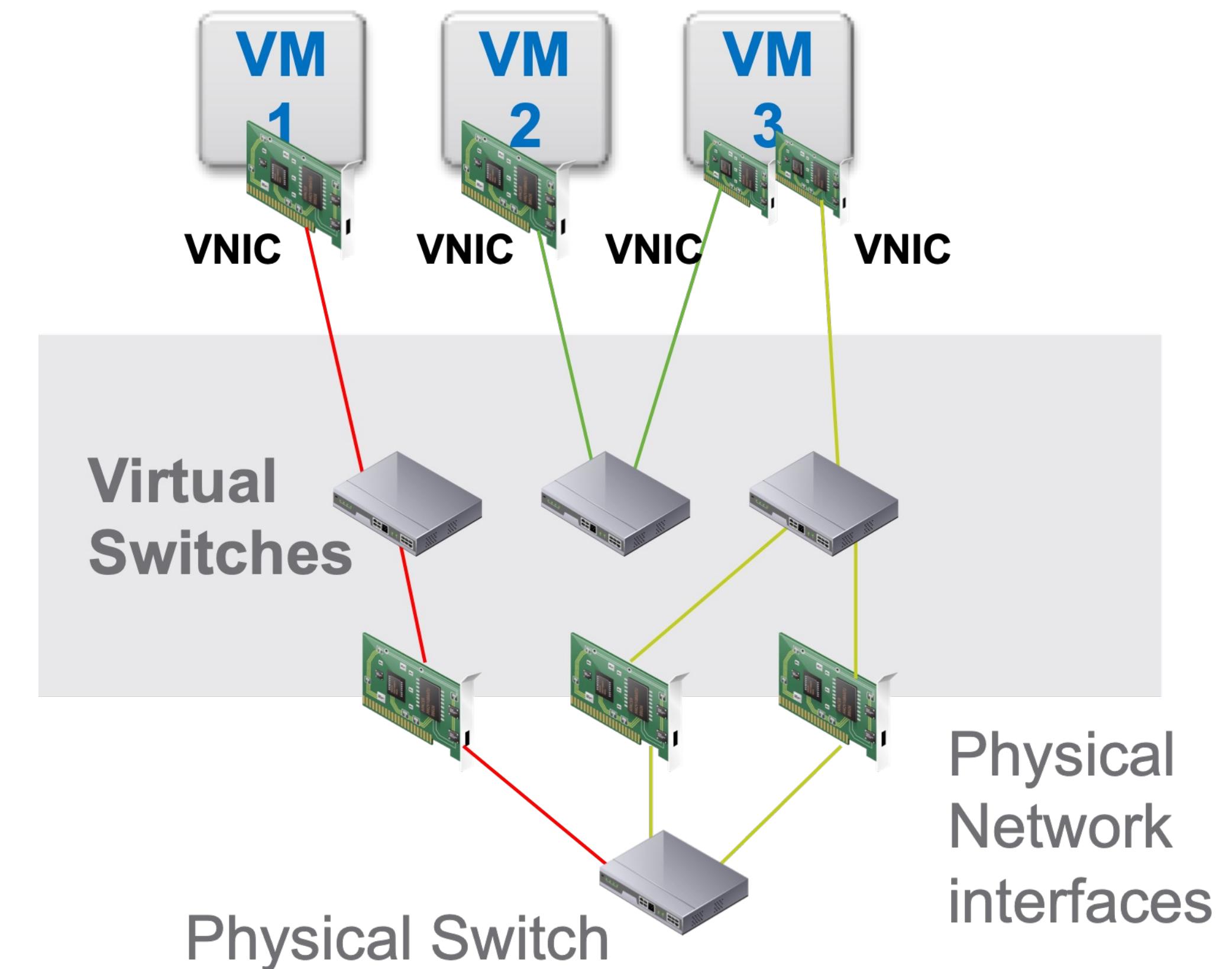
Redes en Virtualización

- Cada máquina virtual puede ser configurada con una o mas NICs.
- El hipervisor simula switching de manera interna para conectar las máquinas virtuales entre sí.
- Múltiples virtual switches pueden ser definidos y configurados para definir que máquinas virtuales se conectan entre sí.
- Al interior del hipervisor la velocidad de conexión entre las máquinas virtuales es controlada por el hipervisor.
- La comunicación máquina a máquina virtual, nunca deja el host.



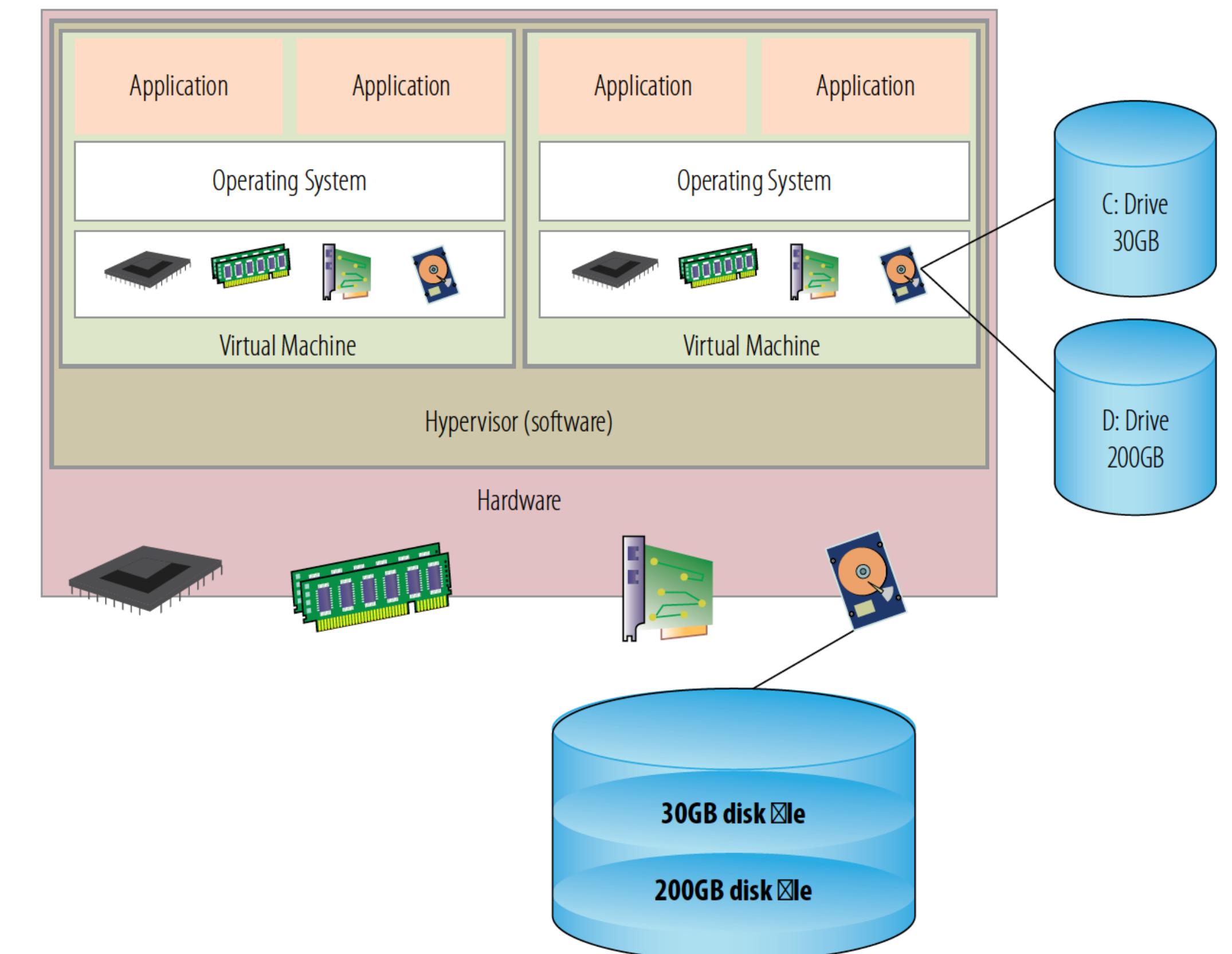
Redes en Virtualización

- Los switches virtuales pueden ser conectados a los switch físicos.
- Para esto los enlace de uplink son configurados utilizando las tarjetas de red físicas en el hipervisor.
- Esto permite a las máquinas virtuales conectarse a la red de datos.



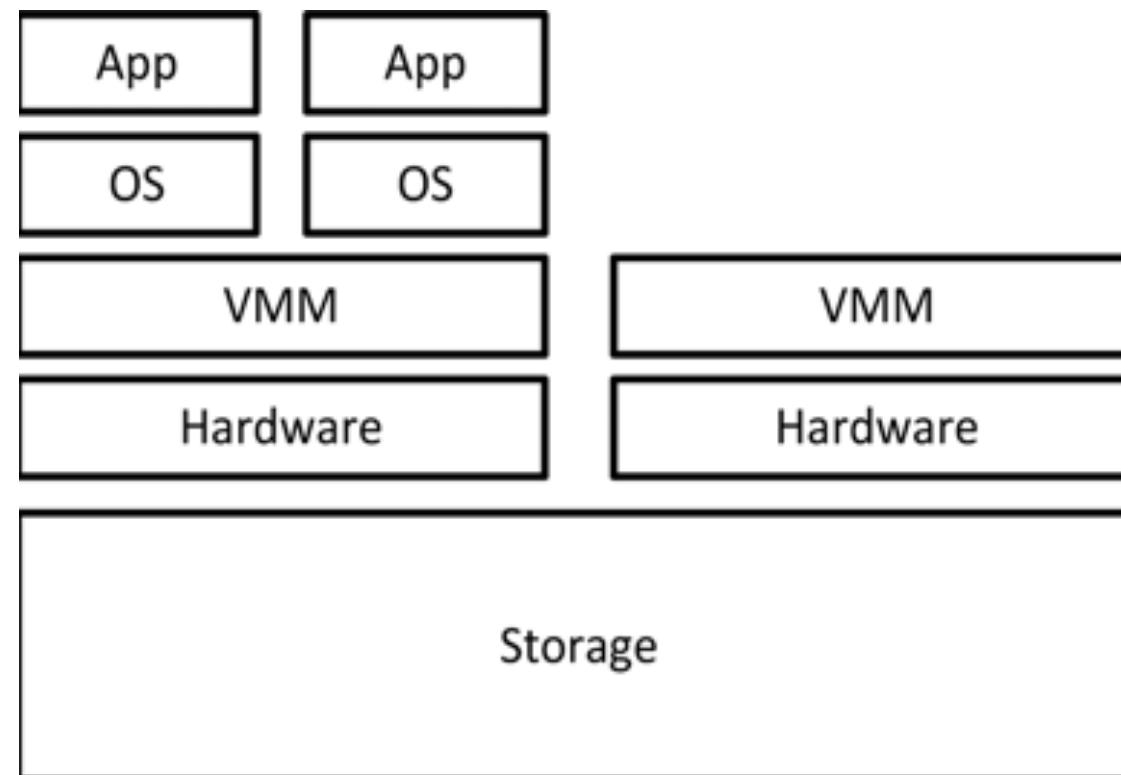
Almacenamiento en Virtualización

- Virtualización a nivel de bloque.

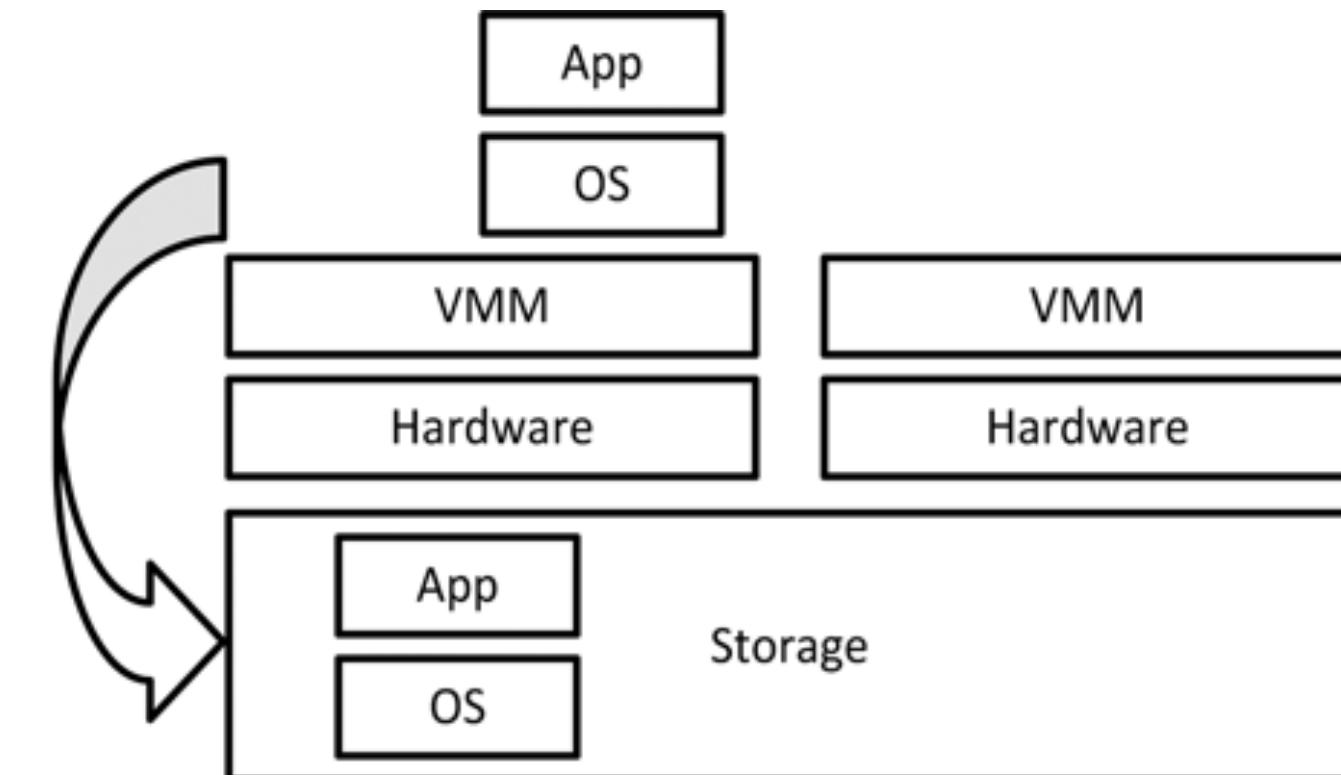


Virtualization Operations

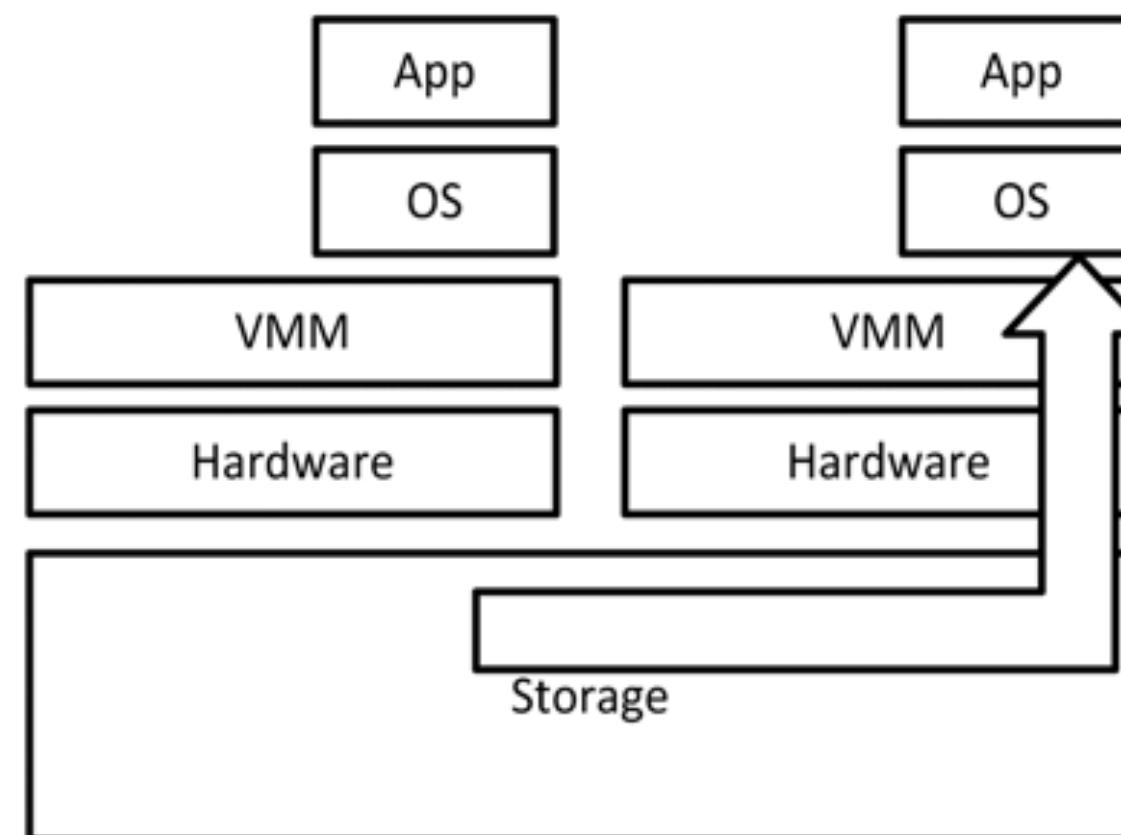
- Un máquina virtual puede multiplexarse entre máquinas de HW.
- Una máquina virtual puede ser suspendida y almacenada en un almacenamiento estable.
- Una máquina suspendida, puede ser aprovisionada en un nuevo HW.
- Una máquina virtual puede ser migrada de una plataforma HW a otra.



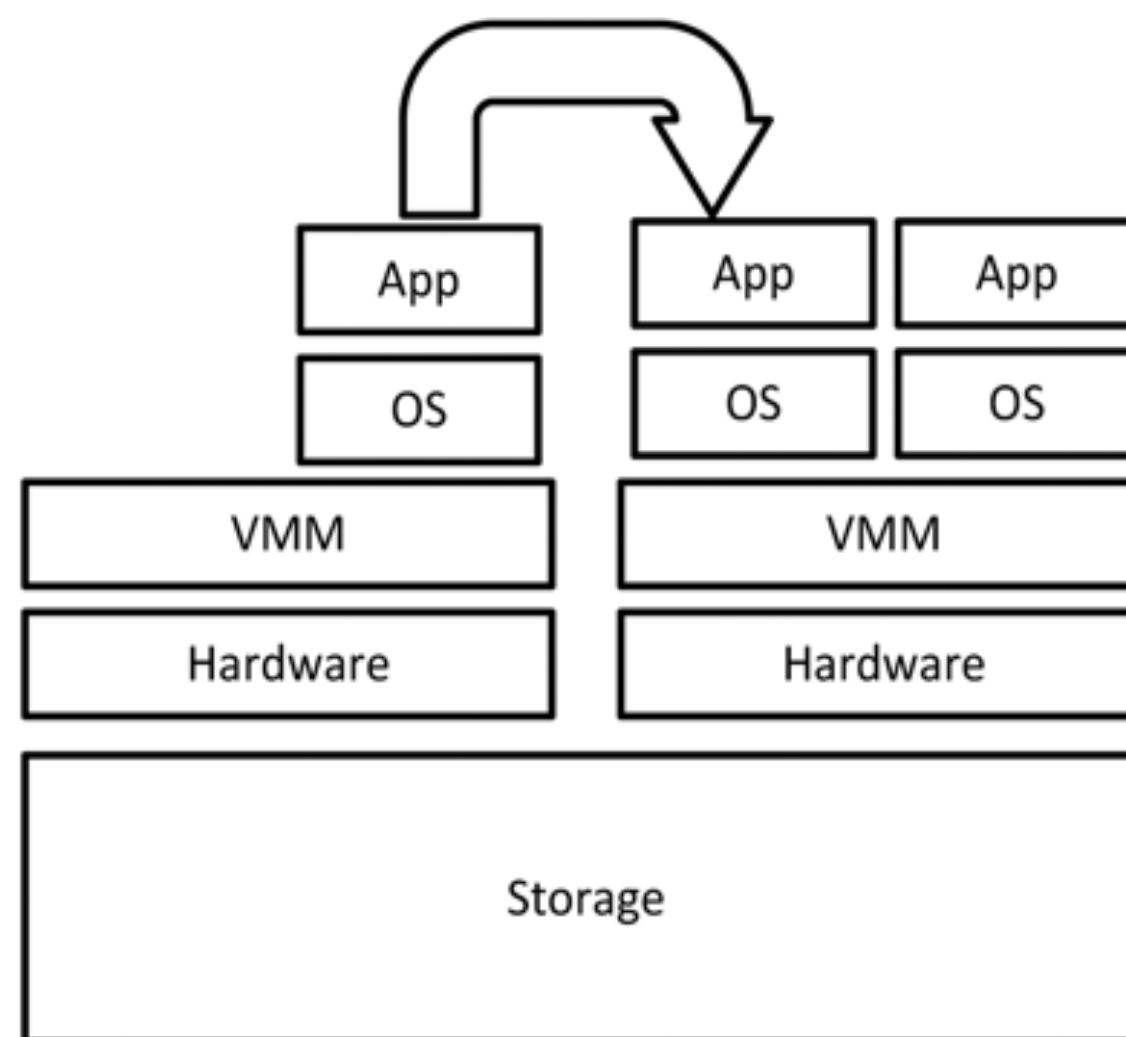
(a) Multiplexing



(b) Suspension (Storage)



(c) Provision (Resume)



(d) Life Migration

Contenedores

Contexto

- Existe un gran movimiento en la actualidad de mover el despliegue de aplicaciones de máquinas virtuales hacia contenedores...
- Dentro de las principales razones se pueden resaltar:
- Flexibilidad y bajo costo en comparación con las máquinas virtuales.
- La popularidad de los contenedores esta creciendo de manera exponencial.
- A medida que se adopten y se utilicen mas contenedores, se hace necesario gestionarlos así como orquestarlos.

Definición

- Un contenedor es una forma de empaquetar una aplicación y todas sus dependencias de tal forma que esta pueda ser movida entre los entornos y se pueda ejecutar sin cambios
- De esta forma un contenedor “contiene” una aplicación y sus dependencias tales como librerías y frameworks.
- Los contenedores trabajan de tal forma que las diferencias entre las aplicaciones queda enmarcada de los contenedores...de esta forma
- Múltiples contenedores, pueden ser ejecutados sobre el mismo host.
- Los contenedores pueden desacoplar aplicaciones del sistema operativo.



Beneficios de los Contenedores

- ✓ Agilizar la creación y el despliegue de aplicaciones.
- ✓ Las aplicaciones están desacopladas de la infraestructura.
- ✓ Las imágenes del contenedor de la aplicación se crean en tiempo de compilación en lugar de tiempo de despliegue.
- ✓ Los contenedores se ejecutan independientemente del medio.
- ✓ Apoyar una mayor utilización de recursos.
- ✓ Las aplicaciones se dividen en pequeñas piezas dependientes que pueden ser manejadas dinámicamente.

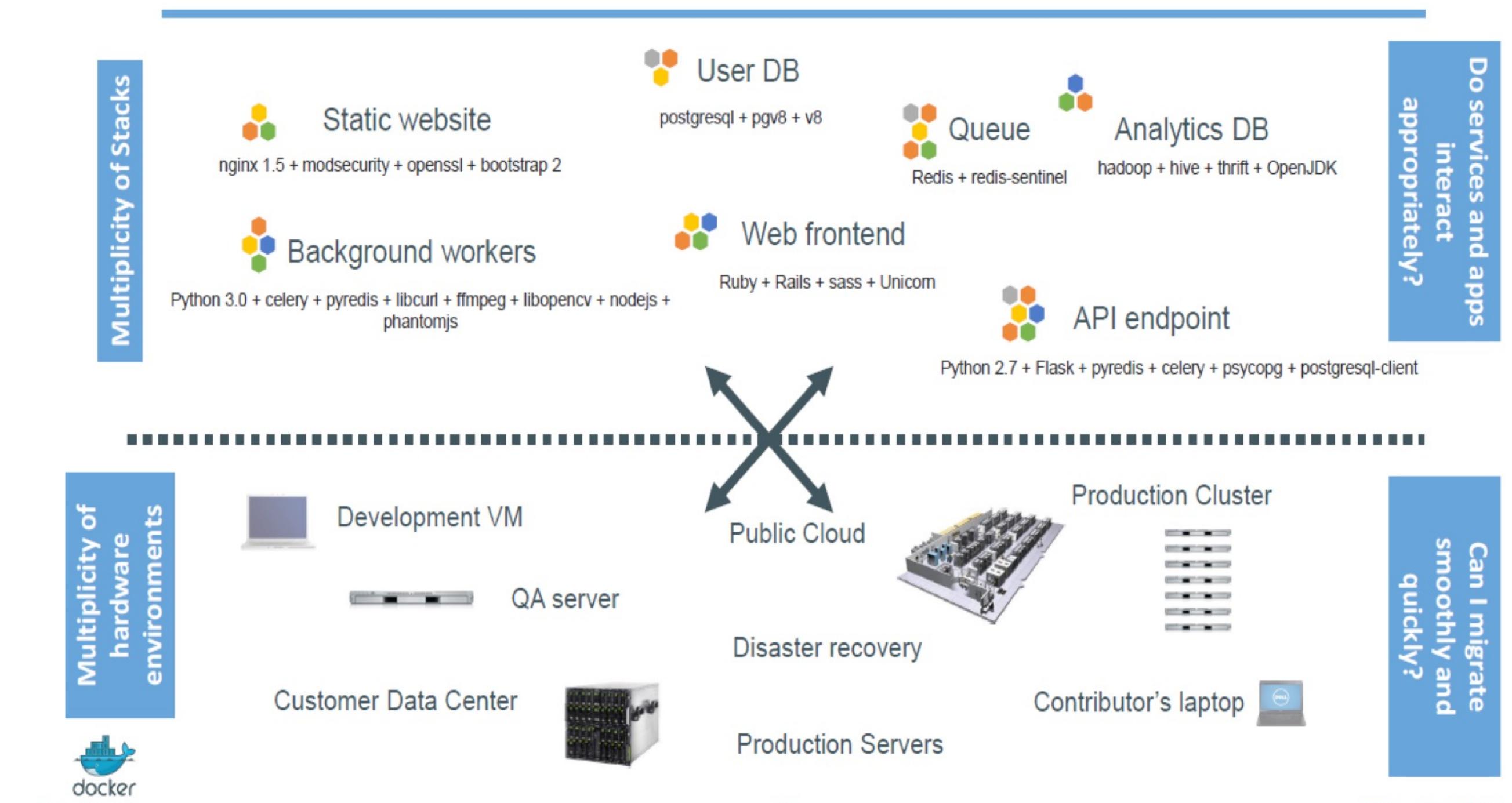
Los contenedores de Docker son:

- ✓ Ligero.
- ✓ Rentable en un entorno de nube pública.
- ✓ Mejor rendimiento.
- ✓ Requiere menos recursos de hardware desde una nube privada.
- ✓ Incrementa productividad.

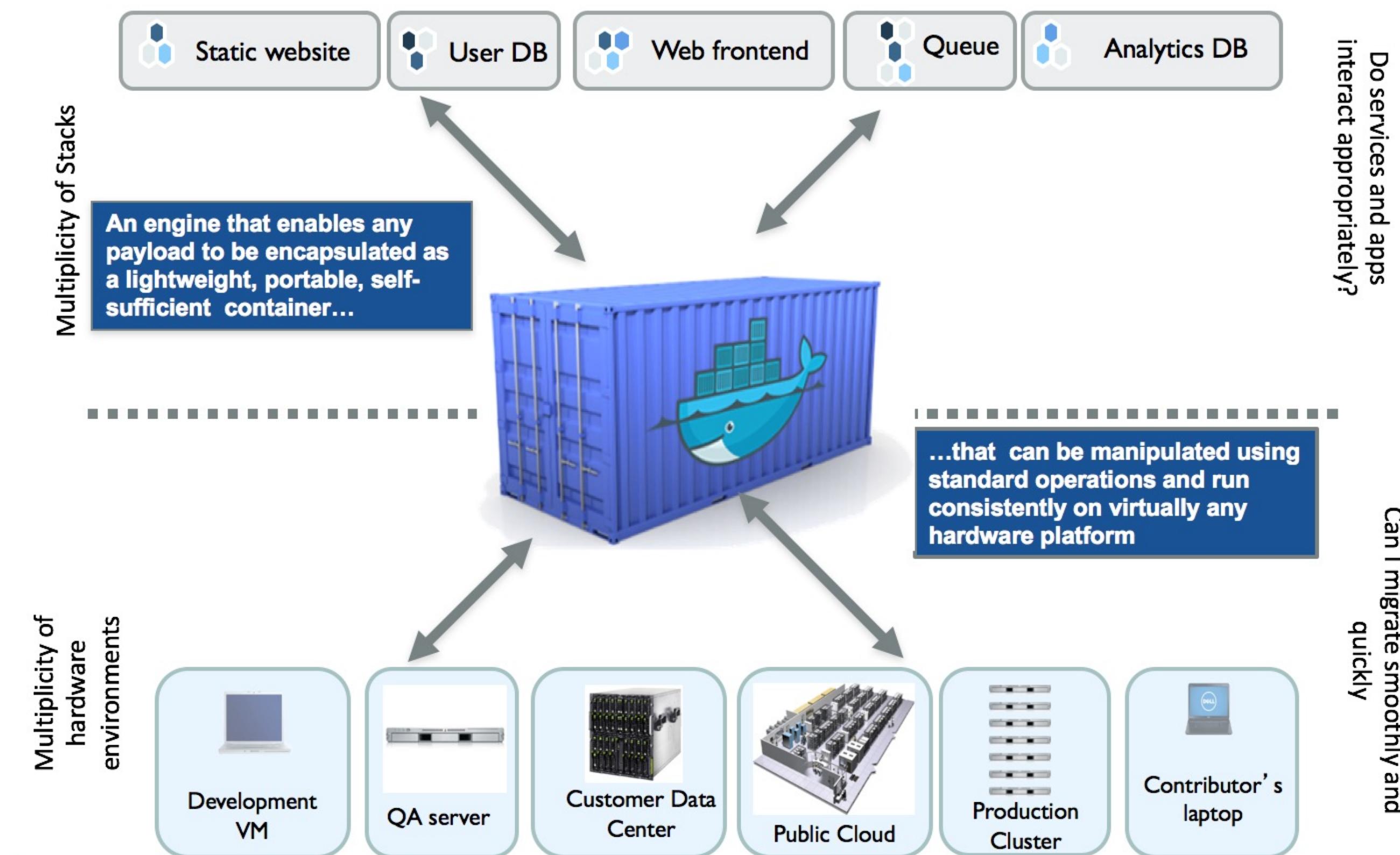


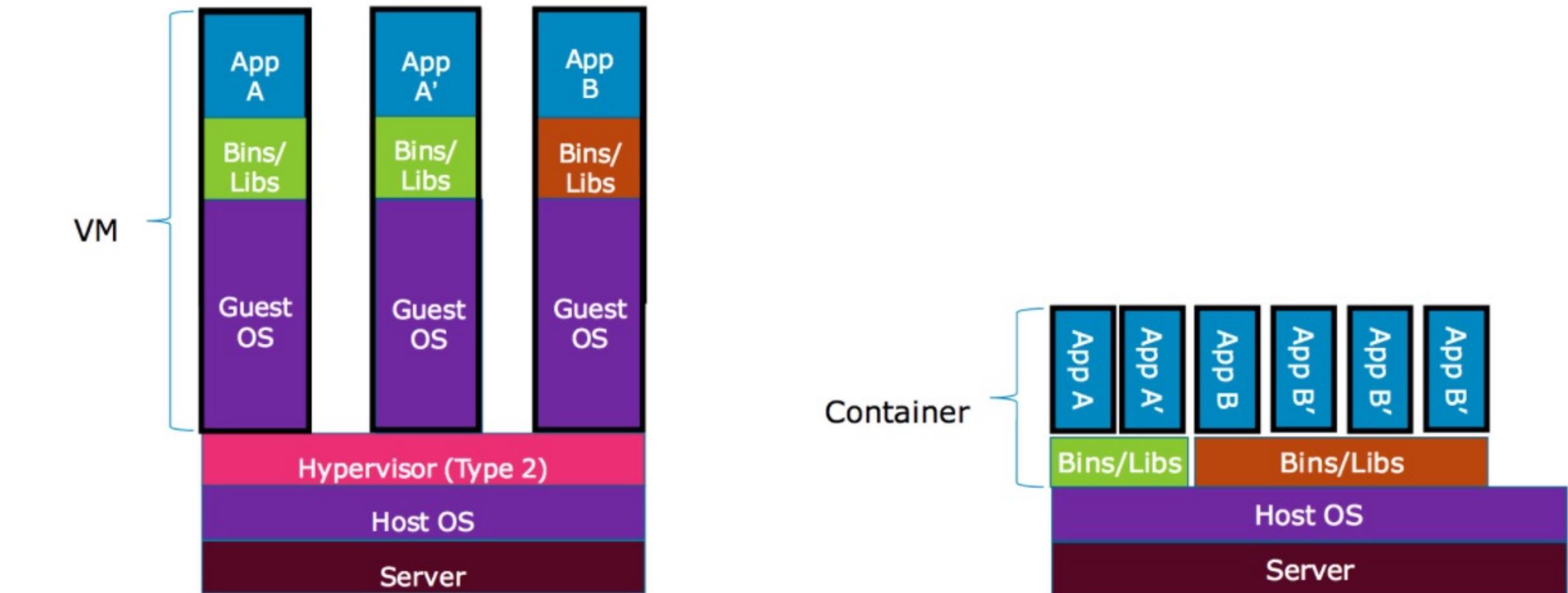
Retos de los Contenedores

- Se basan en la virtualización del sistema operativo en lugar de la virtualización del HW.
- Su fin es abstraer un sistema operativo e incluir aplicaciones o tareas, así como todas sus dependencias.
- Los contenedores son objetos portátiles e independientes que pueden ser fácilmente manipulados por la capa de software que administra una gran computadora virtual.

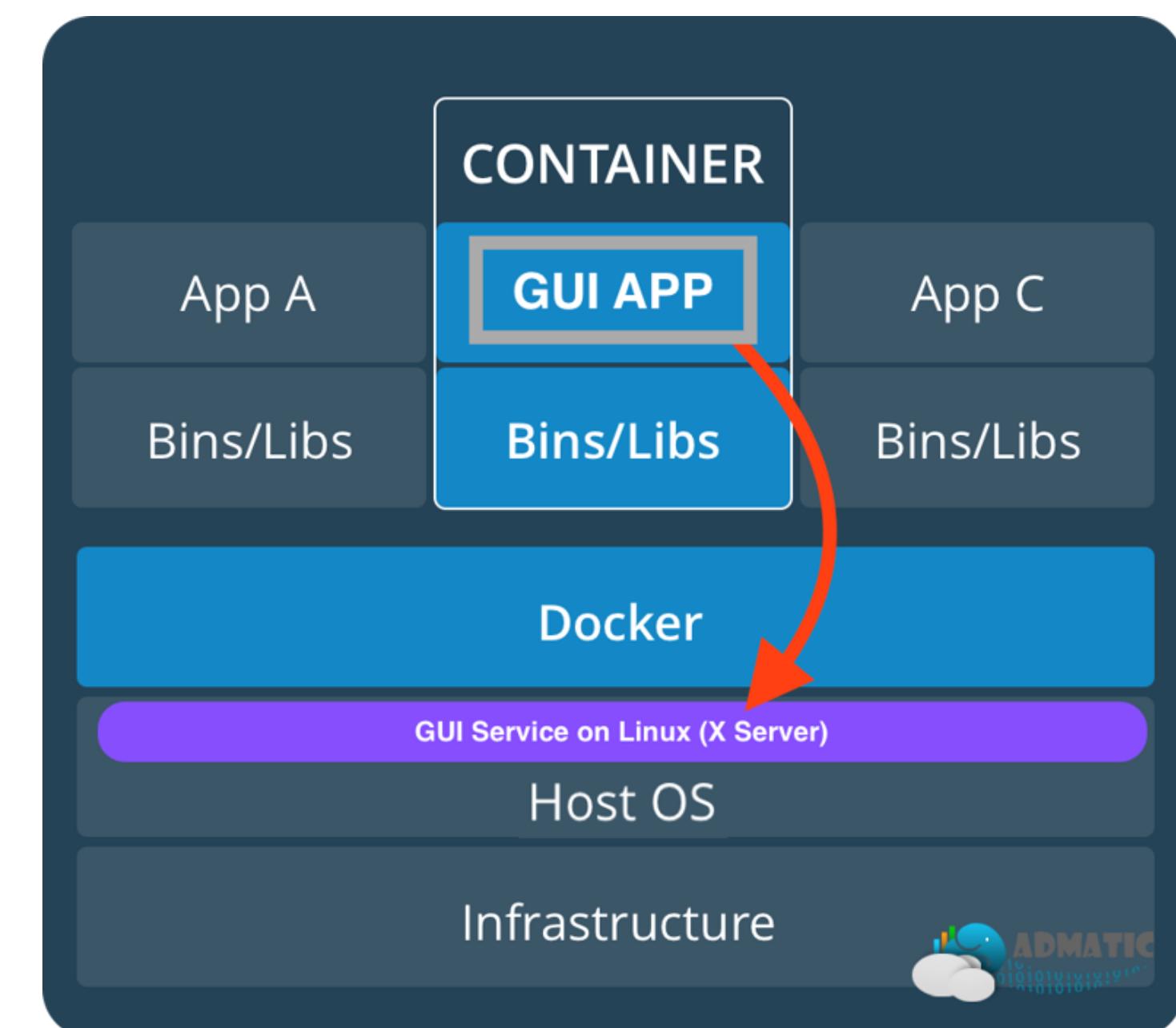


Permite a los desarrolladores elegir los lenguajes de programación y los sistemas de software más adecuados y elimina la necesidad de hacer copias del código de producción e instalar la misma configuración en diferentes entornos.





Máquinas Virtuales vs Contenedores



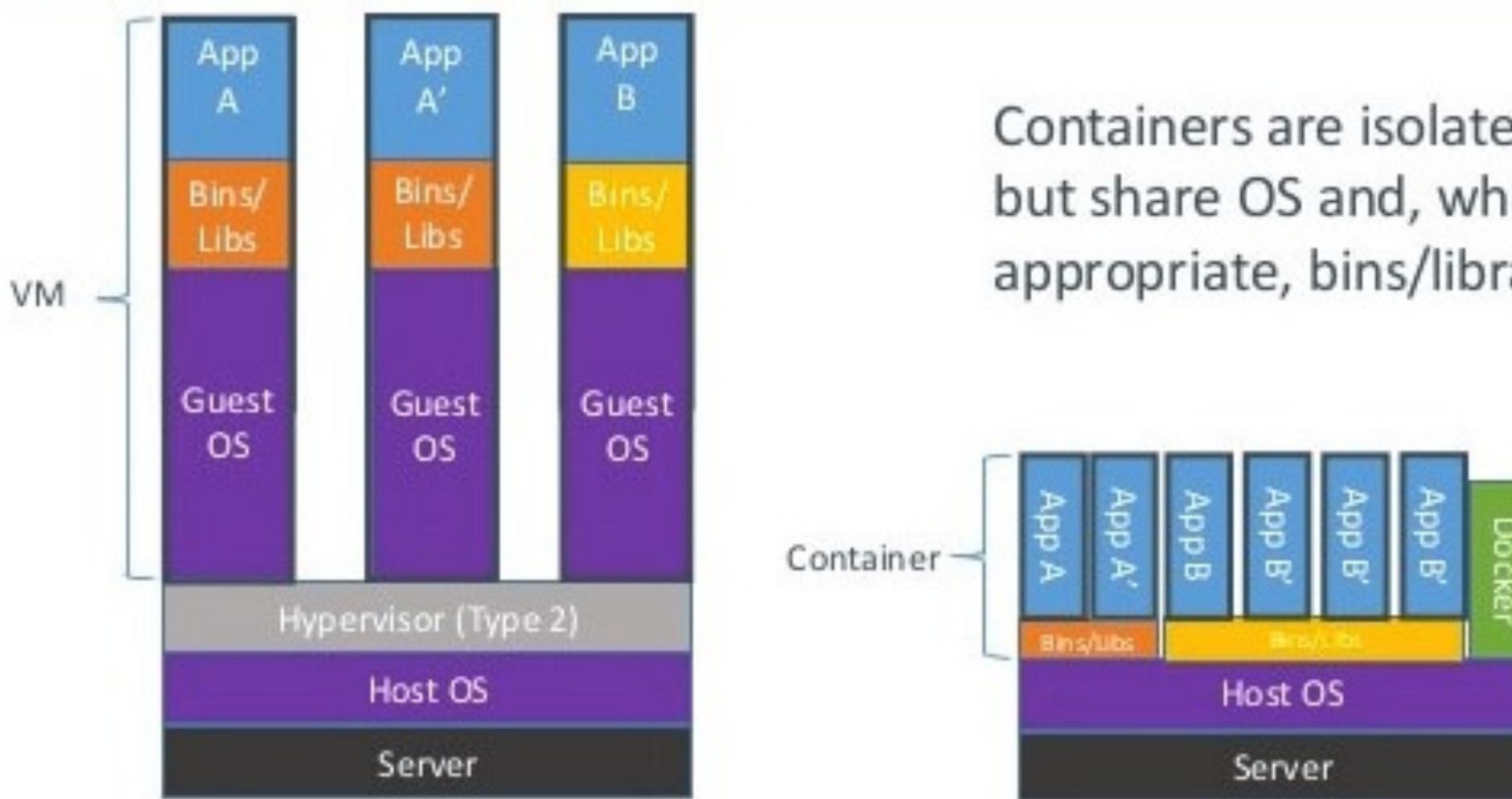
Casos de Uso para los Contenedores

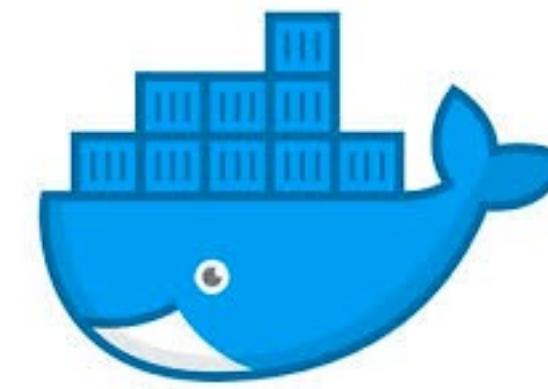
- Combinación de Arquitectura Microservicios/Contenedores
 - Servicios débilmente acoplados...
 - Servicios autónomos y desplegados independientemente...
- DevOps
 - Desarrollar, distribuir y ejecutar SW.
- On-premise, Cloud, Hybrid y Multi-Cloud.
 - Contenedores se ejecutan en diferentes entornos...
- Modernización de aplicaciones...

Virtualización

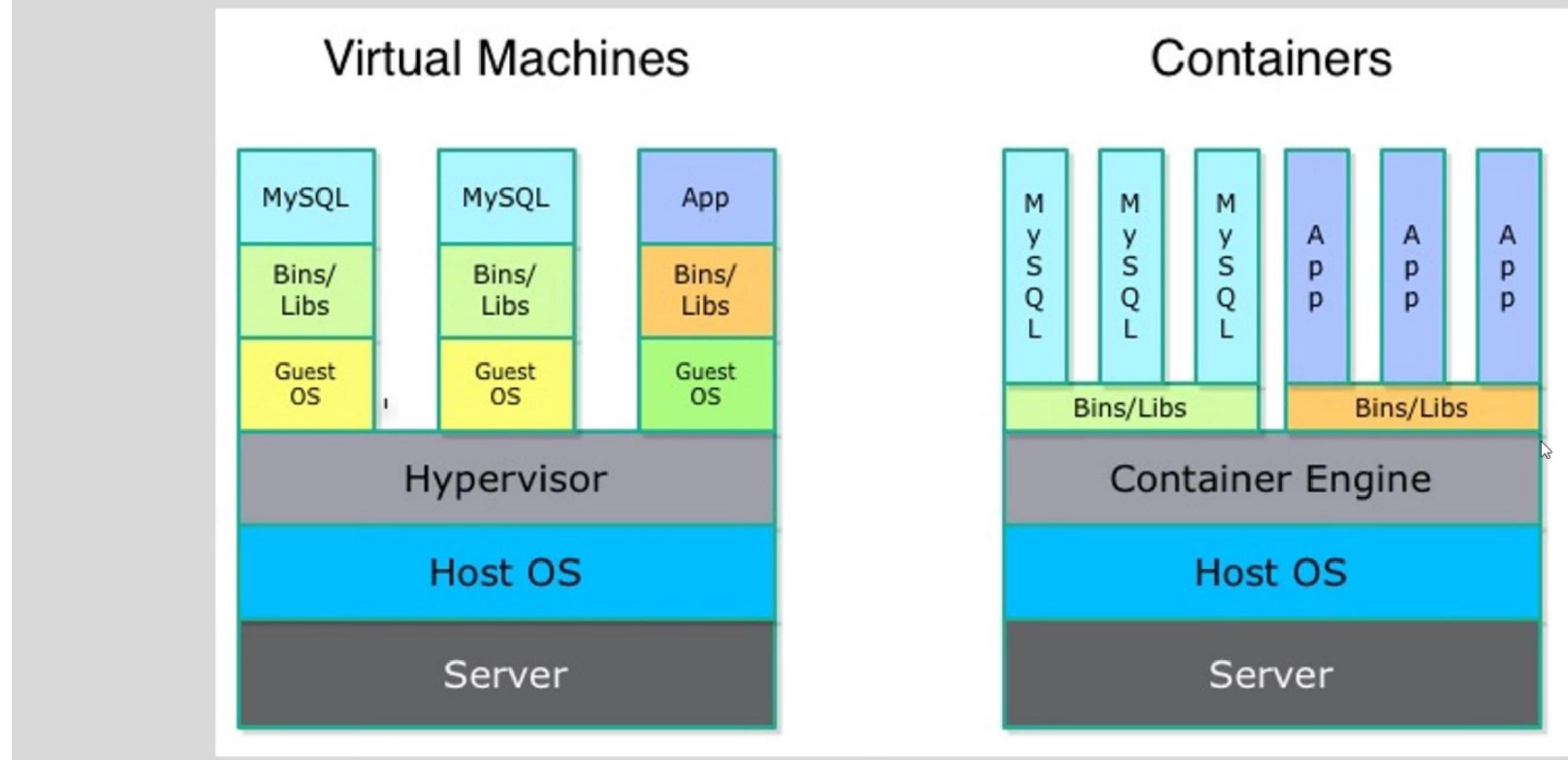


Containers vs. VMs

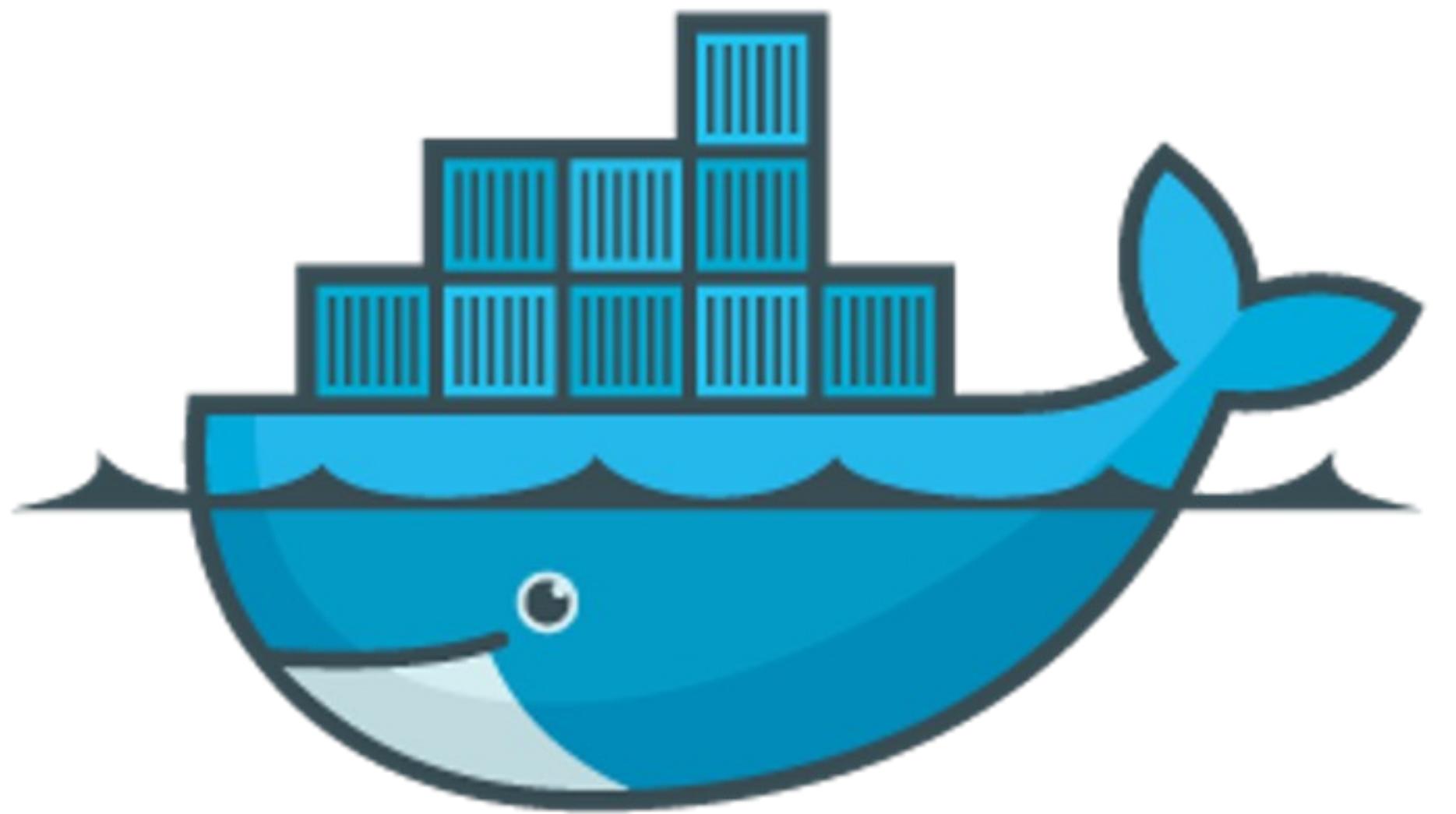




Virtual Machine Vs Containers

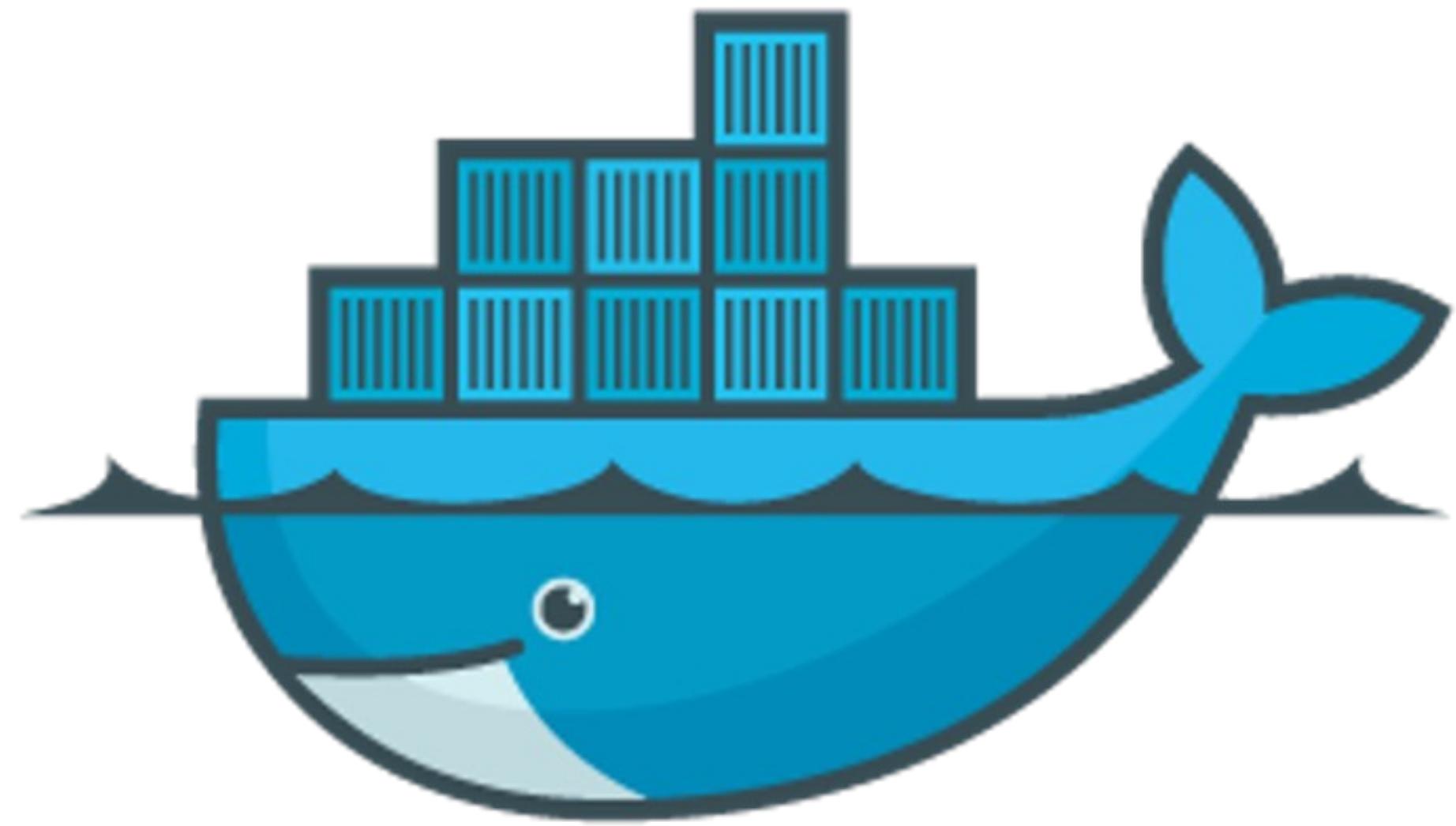


Docker



¿Qué es Docker?

- Docker es un proyecto open source para la creación, distribución y ejecución de aplicaciones software como contenedores.
- Fue lanzada en 2013...
- Dos conceptos importantes en docker:
 - Imagen...
 - Contenedor...
- Simula los contenedores reales, en los cuales un Contenedor esta aislado.
- Docker NO utiliza virtualización persé.
- Utiliza las librerías de Linux para Aislar un proceso completamente.



Title

Life before Docker

Three times the effort to manage deployment

Install, configure, and maintain complex application
↓
Dev laptop Test server Live server

Life with Docker

A single effort to manage deployment

Install, configure, and maintain complex application
↓
Docker image → Dev laptop
Docker image → Test server
Docker image → Live server

- Docker emplea una arquitectura Cliente/Servidor.
- Cliente y daemon pueden co-existir en la misma máquina.
- Los clientes se comunican con el servidor a través de API REST, sockets, interfaz de red.

Docker client:

- Es la forma principal con el usuario se comunica con el motor de docker...
- Los comandos son enviados a través de la API al daemon de docker, el cual los ejecuta.

- Los clientes son CLI y docker compose.

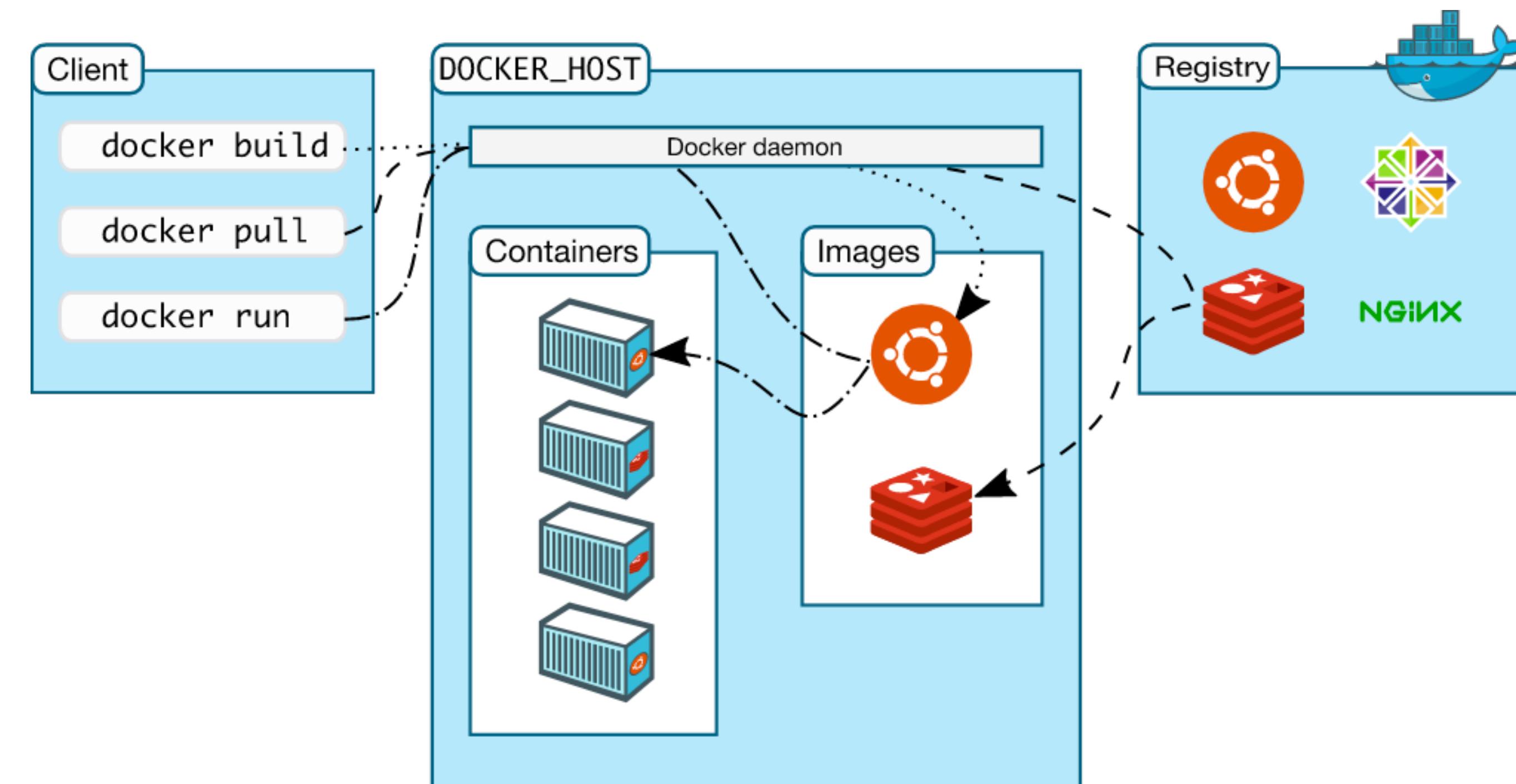
Docker daemon:

- Es el proceso que escucha las peticiones del cliente y ejecuta la tarea.
- Se encarga de la gestión de los objetos: imágenes, contenedores, redes, volúmenes, etc.

Docker registries:

- Es el encargado de almacenar las imágenes de docker.
- El docker hub es registro público que cualquiera puede usar. Docker es configurado por defecto para utilizar el docker hub

Arquitectura Docker



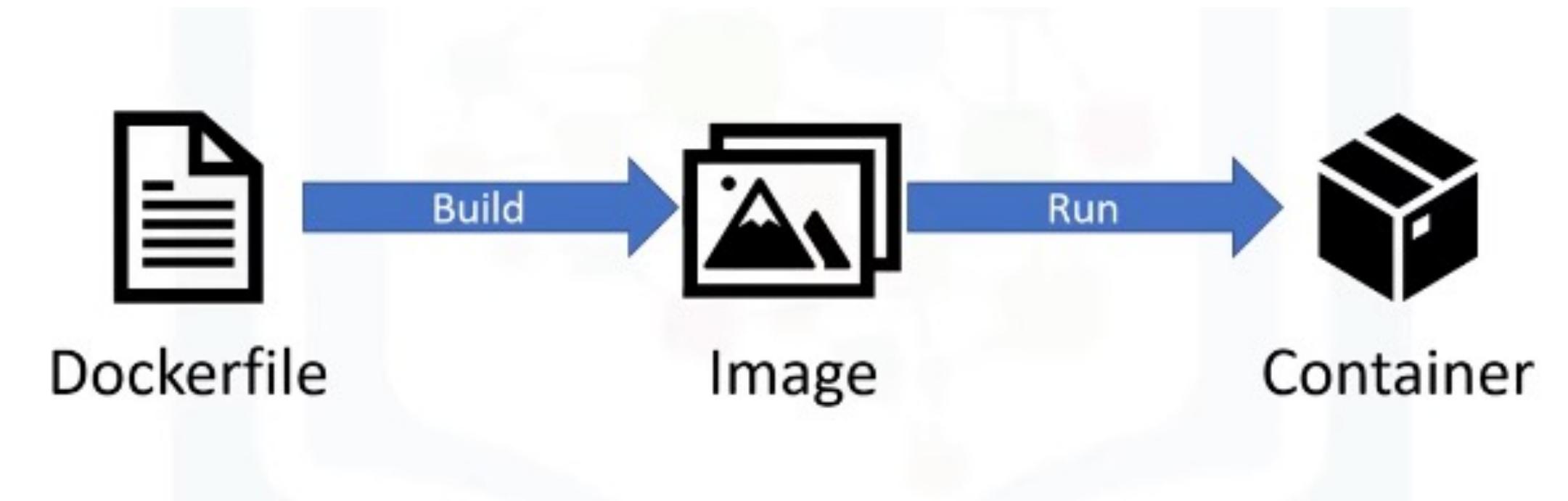
Objetos en Docker

- **Imágenes:**

- Una imagen se define como un template (inmutable) de solo lectura para la creación de un contenedor.
- Normalmente, usted crea una imagen a partir de otra...y agrega algunos aspectos propios que usted requiera.
- Usted puede crear su imagen a través de un archivo que se denomina Dockerfile.
- Una imagen contiene todo lo necesario para ejecutar su aplicación.

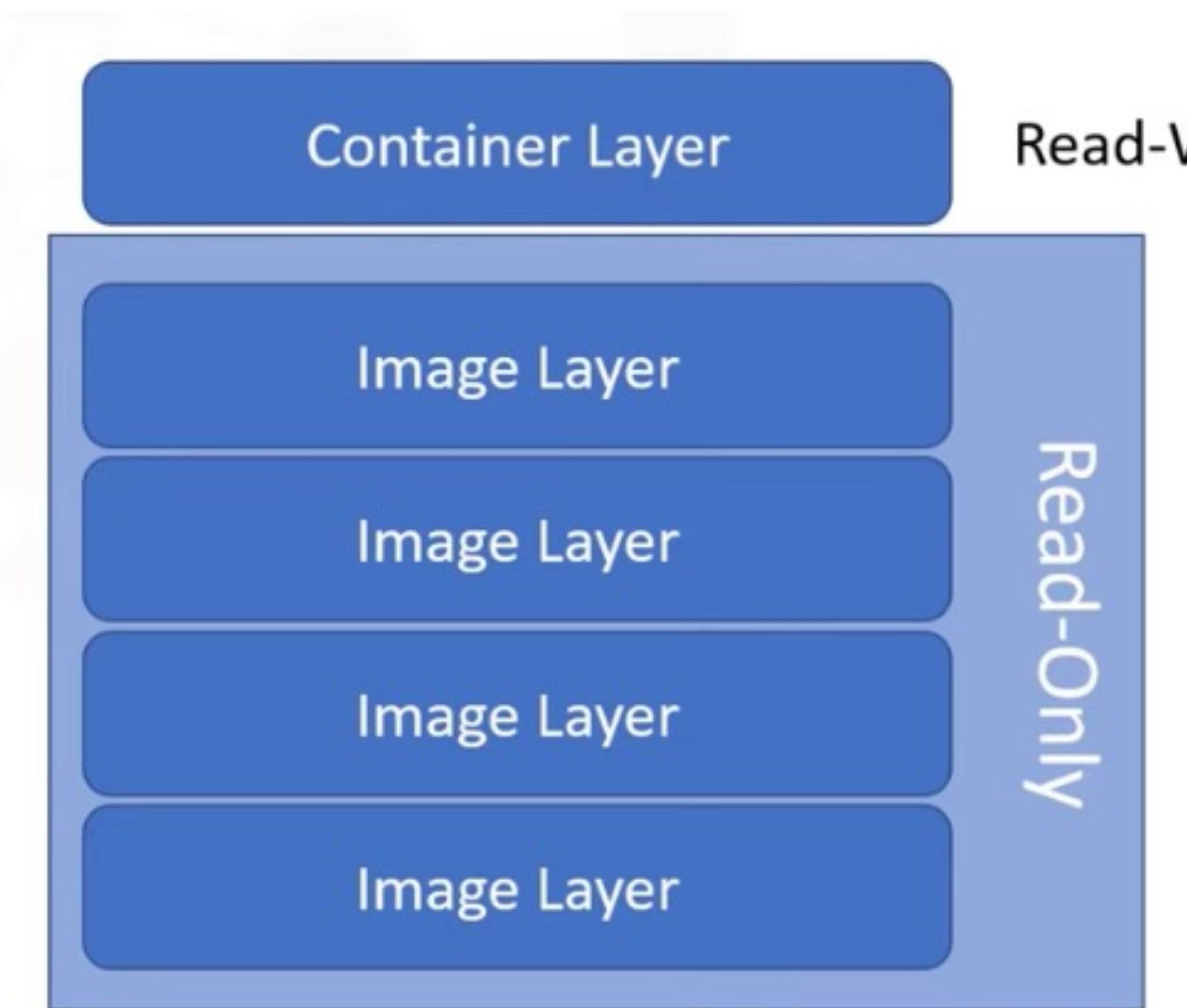
- **Contenedores:**

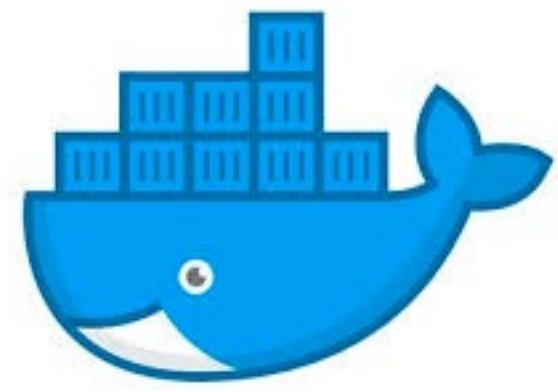
- Un contenedor es una instancia en ejecución de una imagen.
- Usted puede crear, iniciar, parar, mover, borrar un contenedor a través de CLI.
- Un contenedor se puede conectar a una o mas redes, se puede asociar a un almacenamiento.
- En términos generales, un contenedor se encuentra bien aislado de otros contenedores.
- Tenga en cuenta que cuando usted quita un contenedor, cualquier cambio al estado que no se almacenó en el almacenamiento persistente, se pierde...



Construyendo una imagen...

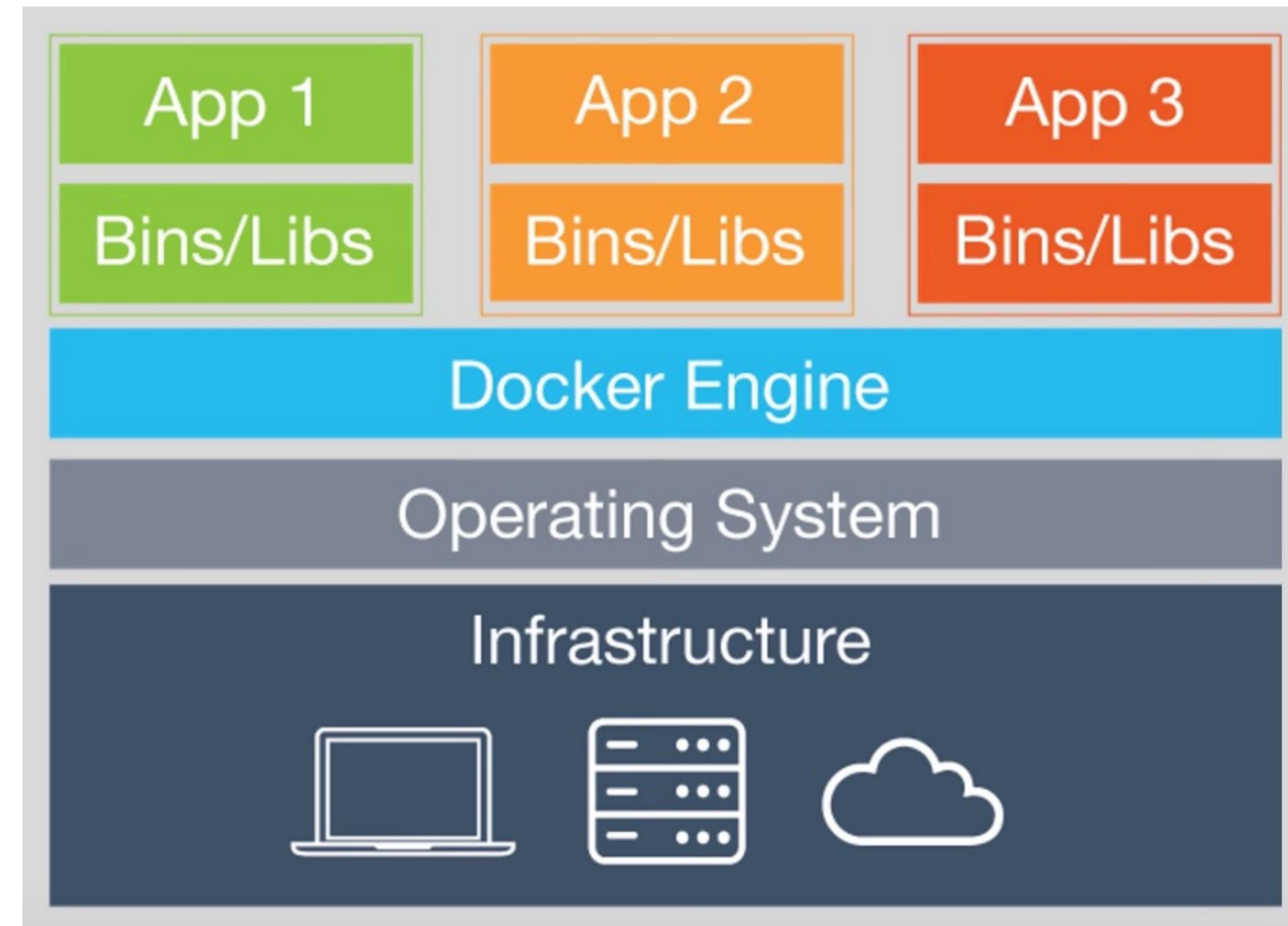
- Las imágenes son construidas con el Dockerfile.
- Un Dockerfile es un archivo texto que contiene todos los comandos que un usuario utiliza para crear la imagen.
- Cada instrucción/comando de docker crea una nueva capa encima de la otra.





- Es una tecnología que nos permite correr un proceso totalmente aislado.
- Caracterizado por:
 - Portabilidad
 - Inmutabilidad
 - ligero

Arquitectura de Docker



Características



- Autogestión de contenedores
- Fiabilidad
- Aplicaciones libres de dependencias del anfitrión
- Múltiples contenedores en un mismo host
- Despliegue en segundos
- Contenedores livianos
- Muchos tipos de aplicaciones, desde webapp hasta SO
- Compatibilidad multisistema y múltiples infraTI
- Se comparten las imágenes en un repo de docker pub / y privado.

Fundamentos de los contenedores

- “virtualización” basada en contenedores que utilizan en kernel de Linux.
- Cada instancia huésped es llamada Contenedor
- Cada contenedor tiene :
 - Su propio sistema de archivos raíz
 - Procesos
 - Memoria
 - Dispositivos
 - Puertos de red
- Utiliza espacios de nombre de linux

- EL Docker Engine es el programa que permite que los contenedores sean construidos, empaquetados y ejecutados
- EL Docker engine usa los espacios de nombre y el control de grupos de Linux para el aislamiento.

Docker comandos básicos

- Instalar docker en tu SO preferido.
- Buscar imágenes:
 - Docker search o ir a hub.docker.com
- # docker pull <image> (traer de hub.docker.com)
- # docker push <image> (envio a hub.docker.com)

Docker-compose

- Permite relacionar 2 o más contenedores que tienen dependencia entre ellos.
- La comunicación se da a través de red. NO de memoria NI archivos.
- Sigue una sintaxis sencilla. Ejemplo:

wordpress

- Requiere: una base de datos (mysql) y la app wordpress:

```
version: '3.3'

services:
  db:
    image: mysql:5.7
  volumes:
    - db_data:/var/lib/mysql
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: somewordpress
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpress
    MYSQL_PASSWORD: wordpress
```

```
wordpress:
  depends_on:
    - db
  image: wordpress:latest
  ports:
    - "8000:80"
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: wordpress
  volumes:
    db_data:
```

Orquestadores Docker

- Docker Swarm
- Kubernetes de google
 - <https://kubernetes.io>
 - Se puede instalar una sandbox local para probar kubernetes.
 - <https://kubernetes.io/docs/setup/minikube/>
 - <https://kubernetes.io/docs/tasks/tools/install-minikube/>

Recursos para aprender Docker

Getting Started Walk-through for IT Pros and System Administrators

Learn more about Docker, how it works and how it can help you deploy secure, scalable applications and save money along the way.

- [Stage 1: The Basics](#)

Learn more about the core concepts of Docker and what it can do for your operations team, and help you understand the fundamental value proposition for Docker. Topics include:

- Fundamentals of Docker
- Deploying a multi-service application

- [Stage 2: Digging Deeper](#)

This stage will help you learn more about some of the advanced topics of Docker. Topics include:

- Security
- Networking
- Orchestration

- [Stage 3: Moving to Production](#)

This stage will give you the resources to deploy a production application, developer a strategy for integrating Docker into your production environment, and get recognized as a leader in your organization on implementing Docker. Topics include:

- Reference Implementations
- Portability
- Storage
- Production anti-patterns

<https://training.play-with-docker.com/>

Recursos para aprender

- <https://training.play-with-docker.com/>
 - <https://youtu.be/M7ZBF-JJWVU>
 - <https://training.play-with-docker.com/ops-s1-hello>
 - <https://training.play-with-docker.com/ops-s1-images>
 - <https://training.play-with-docker.com/ops-s1-swarm-intro>

Docker on AWS

- Docker Swarm via CloudFormation
- Docker simple en una VM (t2.micro) de clase.
- EKS – Elastic Kubernetes Services. Servicio Administrado para crear y gestionar clústeres Kubernetes
- ECS – Elastic Container Service. Son contenedores sin servidor. 100% administrados

Reto

- Conformar un cluster Swarm entre los equipos de proyecto2.
- Utilizar las máquinas ec2 de amazon o vm de gcp
- Utilizar Un (1) manager y n Workers.
- Utilizar como guía:
 - <https://training.play-with-docker.com/orchestration-hol/>
- Desplegar appwebArticulosNodejs de clase como ejemplo.

Docker Instructions

- FROM:
 - Define una imagen base.
- RUN:
 - Ejecuta comandos...
- ENV:
 - Establece variables de entorno...
- ADD/COPY:
 - Son similares y permiten la copia de archivos en la imagen.
- CMD:
 - Proporciona comando para ejecutar el contenedor.

Ejemplo de un Dockerfile

```
FROM ubuntu:22.04
```

```
COPY . /app
```

```
RUN make /app
```

```
CMD python /app/app.py
```