Apuntes Docker

Módulo 1: Introducción a Docker y Contenedores
1.1 ¿Qué es la virtualización de contenedores?
1.2 Ventajas de usar contenedores en el desarrollo y despliegue de aplicaciones.
1.3 Comparación entre máquinas virtuales y contenedores.
1.4 Arquitectura de Docker: imágenes y contenedores.

1.1 La pieza fundamental de la tecnologia Docker es el llamado contenedor o container cuyo funcionamiento se basa en una característica de Linux: Chroot (change root)

Docker es principalmente una empresa creada por Solomon Hykes que ha desarrollado un motor Open Source. La primera versión apareció en 2013.
Docker se desarrolló principalmente como solución a la implementación a sistemas distribuidos, orientado a Cloud.

1.2 docker vs máquinas virtuales: docker usa el kernel compartido. La interacción con usuario se produce a través de diferentes APIs y la línea de comando. Las máquinas virtuales
Las máquinas virtuales (VMs) pueden comunicarse entre sí de varias maneras, similar a cómo lo harían las máquinas físicas en una red. Aquí hay algunas formas comunes en las que las máquinas virtuales pueden establecer comunicación:

diferencia entre docker y VM:

Docker y las máquinas virtuales (VMs) son tecnologías de virtualización, pero difieren en sus enfoques y la forma en que logran el aislamiento y la portabilidad de las aplicaciones. Aquí hay algunas diferencias clave entre Docker y máquinas virtuales:

1. **Nivel de abstracción:**

    - **Docker:** Utiliza contenedores, que son unidades de virtualización a nivel de sistema operativo. Los contenedores comparten el núcleo del sistema operativo subyacente y son más livianos en comparación con las máquinas virtuales.
    - **Máquinas virtuales:** Utilizan hipervisores para crear máquinas virtuales independientes que emulan hardware completo, incluido el sistema operativo. Cada VM tiene su propio sistema operativo y utiliza recursos dedicados.

2. **Consumo de recursos:**

    - **Docker:** Los contenedores comparten el mismo núcleo del sistema operativo, lo que significa que son más eficientes en el uso de recursos. Varios contenedores pueden ejecutarse en la misma máquina sin duplicar el sistema operativo subyacente.
    - **Máquinas virtuales:** Cada VM tiene su propio sistema operativo y consume más recursos, ya que incluye duplicación de sistemas operativos, múltiples kernels y una mayor carga de administración.

3. **Tiempo de inicio:**

   - **Docker:** Los contenedores se inician rápidamente, en cuestión de segundos, ya que no requieren arrancar un sistema operativo completo.
   - **Máquinas virtuales:** Las VMs suelen tardar más en iniciarse, ya que deben arrancar un sistema operativo completo y, en algunos casos, cargar una imagen del sistema operativo desde el disco.

4. **Portabilidad:**

   - **Docker:** Las aplicaciones en contenedores son altamente portátiles y pueden ejecutarse de manera consistente en cualquier entorno que admita Docker, desde el entorno de desarrollo local hasta la nube.
   - **Máquinas virtuales:** Las VMs también son portátiles, pero debido a su mayor tamaño y complejidad, la migración entre diferentes entornos puede ser más desafiante.

5. **Escalabilidad:**

   - **Docker:** Docker facilita la implementación y la escalabilidad de aplicaciones a través de herramientas de orquestación como Kubernetes, Docker Swarm, etc.
   - **Máquinas virtuales:** Aunque las VMs se pueden escalar, la gestión y orquestación de VMs generalmente implica más complejidad.

6. **Aislamiento:**

   - **Docker:** Los contenedores proporcionan un nivel de aislamiento, pero comparten el mismo núcleo del sistema operativo. Es adecuado para la mayoría de las aplicaciones, pero no es tan fuerte como el aislamiento de las máquinas virtuales.
   - **Máquinas virtuales:** Ofrecen un mayor nivel de aislamiento, ya que cada VM tiene su propio sistema operativo y kernel. Esto puede ser preferible para cargas de trabajo que requieren un alto grado de aislamiento.

En resumen, Docker y máquinas virtuales son tecnologías de virtualización que abordan diferentes necesidades y situaciones. Docker es especialmente eficiente para aplicaciones modernas y ágiles, mientras que las máquinas virtuales son más adecuadas para cargas de trabajo que requieren un alto grado de aislamiento y compatibilidad con sistemas operativos diferentes. En muchos casos, estas tecnologías se utilizan de manera complementaria, con contenedores dentro de máquinas virtuales para aprovechar las ventajas de ambas.

Cómo se comunican entre sí las VM

1. **Redes locales o privadas:**
    - **Misma red virtual:** Si las máquinas virtuales están en la misma red virtual o LAN virtual, pueden comunicarse entre sí utilizando direcciones IP dentro de esa red. Esto es similar a cómo funcionaría una red local física.
    - **Bridges de red:** Algunos hipervisores permiten la creación de redes virtuales o bridges que conectan varias máquinas virtuales. Las VMs conectadas a la misma red virtual pueden comunicarse como si estuvieran en una red física compartida.
2. **Redes externas:**
    - **Redes externas o públicas:** Si las máquinas virtuales tienen direcciones IP accesibles desde una red externa (por ejemplo, en la nube), pueden comunicarse entre sí a través de la red externa. Esto implica configurar reglas de firewall y gestión de direcciones IP públicas.
3. **Servidores de nombres (DNS):**
    - **Resolución de nombres:** Las máquinas virtuales pueden comunicarse entre sí utilizando nombres de host en lugar de direcciones IP. Para esto, se puede configurar un servidor de nombres (DNS) que asocie nombres de host con direcciones IP.
4. **Servicios de descubrimiento:**
    - **Servicios de descubrimiento:** Algunos sistemas utilizan servicios de descubrimiento para permitir que las máquinas virtuales se encuentren entre sí dinámicamente. Esto puede implicar el uso de herramientas como Consul, etcd o servicios de registro y descubrimiento de servicios.
5. **Conexiones punto a punto:**
    - **Conexiones punto a punto:** Las máquinas virtuales también pueden establecer conexiones directas entre sí a través de protocolos de red específicos, como TCP/IP. Esto es útil para aplicaciones que requieren comunicación directa sin pasar por una red más amplia.
6. **Redes privadas virtuales (VPN):**
    - **VPNs:** Si las máquinas virtuales están en ubicaciones geográficas diferentes, se pueden conectar a través de una red privada virtual (VPN). Esto proporciona una conexión segura y privada entre las máquinas virtuales, independientemente de su ubicación física.

Es importante tener en cuenta que la configuración específica puede variar según el hipervisor o la plataforma en la que se estén ejecutando las máquinas virtuales. Además, las consideraciones de

seguridad, como la configuración de firewalls y la gestión de accesos, son esenciales para garantizar una comunicación segura y controlada entre las máquinas virtuales.

Cómo se comunican entre sí los docker:
Los contenedores Docker se comunican entre sí a través de redes Docker. Docker proporciona un modelo de red predeterminado que permite a los contenedores comunicarse entre sí de manera sencilla. Aquí hay algunos aspectos clave sobre cómo se logra la comunicación entre contenedores Docker:

1. **Redes Docker:**
    - Docker utiliza el concepto de redes para permitir la comunicación entre contenedores. Puedes crear redes personalizadas o utilizar la red predeterminada llamada "bridge."
2. **Red Bridge (puente) (Predeterminada):**
    - Cuando creas un contenedor sin especificar una red, Docker lo conecta automáticamente a la red "bridge". Los contenedores en la misma red bridge pueden comunicarse entre sí utilizando direcciones IP internas.

```
bashCopy code
# Ejemplo de ejecución de contenedores en la misma red bridge
docker run --name contenedor1 -d nginx
docker run --name contenedor2 -d nginx
```

En este caso, los contenedores `contenedor1` y `contenedor2` pueden comunicarse entre sí a través de la red bridge.

3. **Nombre del Contenedor:**
    - Docker asigna automáticamente un nombre de host basado en el nombre del contenedor. Puedes utilizar este nombre de host para la comunicación entre contenedores en la misma red.

```
bashCopy code
# Ejemplo de comunicación por nombre de contenedor
docker run --name contenedor1 -d nginx
docker run --name contenedor2 -d nginx

# Desde contenedor1, puedes hacer referencia a contenedor2 por su nombre de host
ping contenedor2
```

4. **Redes Personalizadas:**
    - Puedes crear tus propias redes personalizadas para aislar y organizar contenedores. Los contenedores en la misma red personalizada pueden comunicarse entre sí.

```
bashCopy code
# Crear una red personalizada
docker network create mi_red

# Ejemplo de ejecución de contenedores en la misma red personalizada
docker run --name contenedor1 -d --network mi_red nginx
docker run --name contenedor2 -d --network mi_red nginx
```

En este caso, los contenedores `contenedor1` y `contenedor2` pueden comunicarse entre sí a través de la red personalizada `mi_red`.

5. **Exposición de Puertos:**

    - Si un contenedor expone un puerto, otros contenedores en la misma red (o incluso en redes diferentes, según la configuración) pueden acceder a ese puerto.

```
bashCopy code
# Ejemplo de exposición de puertos
docker run --name contenedor1 -d -p 8080:80 nginx
docker run --name contenedor2 -d --network mi_red nginx

# Desde contenedor2, puedes acceder a contenedor1 a través del puerto 8080
curl contenedor1:8080
```

En este caso, `contenedor2` puede acceder al puerto 8080 del `contenedor1`.

6. **Orquestación de Contenedores:**

    - En entornos más complejos y en producción, las herramientas de orquestación como Docker Compose o Kubernetes facilitan la gestión y la comunicación entre contenedores en clústeres.

Estas son algunas de las formas básicas en que los contenedores Docker pueden comunicarse entre sí. Es importante comprender cómo funcionan las redes Docker y cómo se configuran para garantizar una comunicación e