TEST DE CONOCIMIENTOS SOBRE DOCKER

(CADA PREGUNTA VALE 1 PUNTO)

1. ¿Qué es Docker?

Docker es una herramienta open-source de virtualización ligera con la que poder empaquetar entornos y aplicaciones que posteriormente podremos desplegar en cualquier sistema que disponga de esta tecnología.
Extiende de LXC(LinuX Containers) que nos permite crear múltiples sistemas aislados entre sí y sobre el mismo anfitrión.

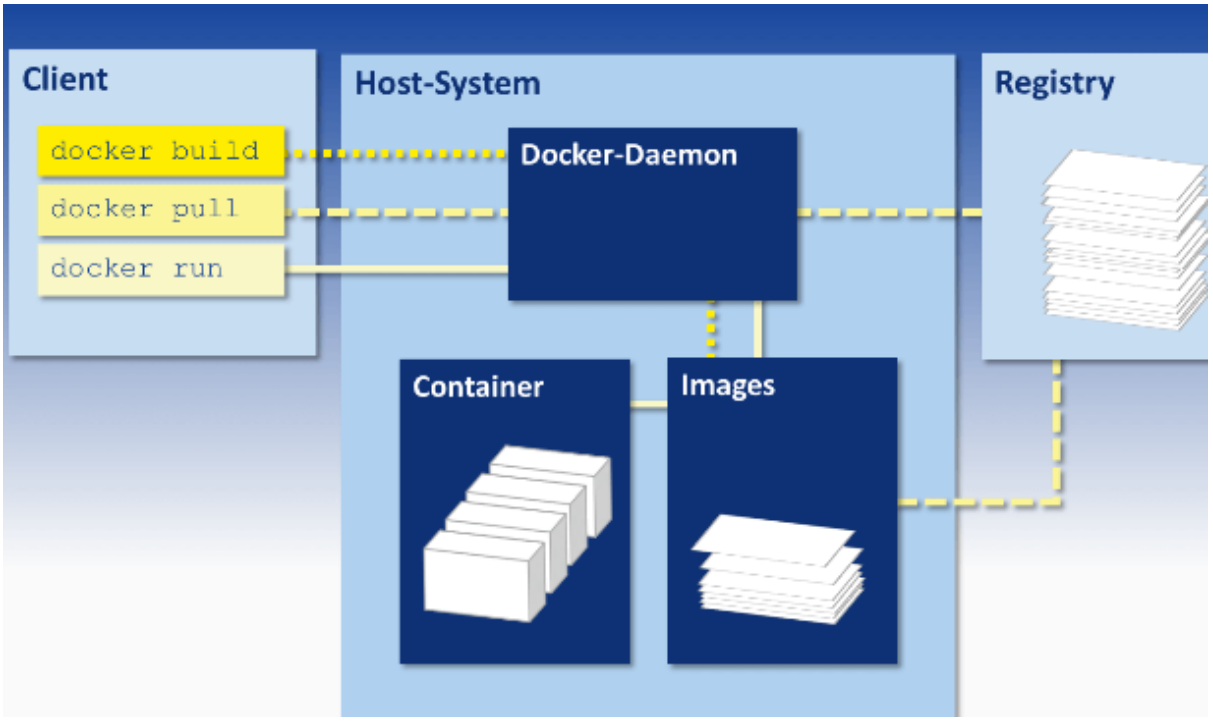
2. ¿Qué son los 'contenedores'?

Los contenedores Docker son las herramientas que Docker usa para empaquetar y enviar las aplicaciones del desarrollador hacia su destino final. Estos contenedores son una función muy valorada de Docker porque pueden ejecutarse en cualquier tipo de máquina - en otras palabras, no son exclusivas a un sistema operativo. La universalidad que proporcionan estos contenedores se vuelve una herramienta valiosa para desarrolladores y programadores.

3. ¿Qué son las 'imágenes' Docker?

Una imagen Docker es un archivo, compuesto por múltiples capas, que se utiliza para ejecutar código en un contenedor Docker. Estas imágenes son las plantillas base desde la que partimos ya sea para crear una nueva imagen o crear nuevos contenedores para ejecutar las aplicaciones.
Las imágenes son utilizadas para crear el contenedor en Docker. Esto se logra emitiendo el comando ¨**run**¨.

4. Explica los conceptos de la siguiente imagen:



El comando docker build da instrucciones al daemon para crear una imagen (línea punteada), para lo cual debe estar disponible el Docker file correspondiente. Si el usuario no ha creado la imagen, sino que la toma de un repositorio en Docker Hub, entonces se ejecuta el comando docker pull (línea discontinua). Cuando se le ordena al daemon iniciar un contenedor con la orden docker run, el programa comprueba primero si la imagen requerida está almacenada de forma local. En caso afirmativo, el contenedor se inicia (línea continua). También puede ocurrir que el daemon no encuentre la imagen, a partir de lo cual extrae una directamente del repositorio.

5. ¿Es confiable la 'tecnología de contenedores'?

Sí, ya que docker permite entregar código con mayor rapidez, estandarizar las operaciones de las aplicaciones, transferir el código con facilidad y ahorrar dinero al mejorar el uso de recursos. La sintaxis sencilla y simple le aporta un control absoluto.
Y si las más grandes compañias del mundo como Google, Amazon, Intel… usan y confian en la tecnología de contenedores, será porque realmente si es confiable.

6. ¿Perderás todo tu trabajo si accidentalmente sales de un contenedor?

No, no perderás ningún tipo de información, datos u otros parámetros si accidentalmente sales del contenedor Docker. La única manera de perder tu avance sería un problema con un comando específico para eliminar el contenedor. Salirse no afectará para nada a los archivos

7. Explica qué hacen las siguientes lineas de comando:

$ sudo docker run hello-world
Pone en marcha un contenedor que se llama hello-world

$ sudo apt-get purge docker-engine
Elimina los contenedores vacios. Limpia los registros

$ sudo systemctl status docker
Muestra el estado del contenedor de docker

$ docker run [OPTIONS] IMAGE [:TAG|@DIGEST] [CMD] [ARG...]
Es la forma general de como usar el comando run, que es la forma de poner en marcha un contenedor

8. ¿Cuáles son los principales factores que imponen el número de contenedores que puedes ejecutar?

En realidad no hay un límite claramente definido de contenedores que puedes correr con Docker. Sin embargo, dicho esto, las limitaciones comienzan cuando hablamos de hardware.

Hay básicamente dos factores que pueden limitar el número de contenedores que puedes ejecutar - el tamaño de tu aplicación y el poder de tu CPU. Si tu aplicación no es enorme y tienes un suministro ilimitado de poder para tu CPU, probablemente puedes ejecutar una gran cantidad de contenedores Docker al mismo tiempo

9. ¿Cuál es el mejor lugar para encontrar buenos ejemplos de 'archivos compose'?

Muchas de las compañías de alto nivel que requieren de expertos en Docker (o aspirantes a expertos - depende) usan una herramienta específica para administrar su trabajo interno. Esta herramienta se llama **GitHub.**

Además de las otras funciones que realiza, es un excelente lugar para encontrar los mencionados archivos compose para contenedores Docker. Se recomienda que presentes GitHub cómo tu respuesta principal a la pregunta - es posible que esa sea exactamente la respuesta que tus empleadores están buscando.

10. Contenedor frente a máquina virtual. Razona tu respuesta.

Cada uno viene con beneficios e inconvenientes. En un entorno VM, cada carga de trabajo necesita un sistema operativo completo. Pero con un entorno contenedor, se pueden ejecutar múltiples cargas de trabajo con 1 SO. Cuanto mayor sea la huella del sistema operativo, más beneficios medioambientales tendrán los contenedores. Con esto, trae más beneficios como recursos de administración de TI reducidos, tamaño reducido de instantáneas, aplicaciones de rotación más rápidas, actualizaciones de seguridad reducidas y simplificadas, menos código para transferir, migrar y cargar cargas de trabajo.

Puede usar Docker para aislar aplicaciones individuales y máquinas virtuales para aislar sistemas completos. Están operando en diferentes niveles de abstracción.

Iniciar una máquina virtual es más costoso en términos de tiempo que iniciar un contenedor.

Son similares en que ambos proporcionan entornos aislados: ambos se pueden usar para empaquetar y distribuir software. Sin embargo, los contenedores suelen ser mucho más pequeños y rápidos, lo que los hace mucho más adecuados para ciclos de desarrollo rápidos y microservicios. La desventaja es que los contenedores no hacen una virtualización verdadera; No puede ejecutar un contenedor de Windows en un host Linux, por ejemplo.

Los contenedores y Docker no están en conflicto con las máquinas virtuales; son tecnologías complementarias para usos distintos. Las máquinas virtuales permiten a los usuarios administrar hosts por API y ofrecen elasticidad de infraestructura. Docker permite a los usuarios definir el software como pequeños bloques de lego para ensamblar, por lo que adoptan arquitecturas modernas: infraestructuras inmutables, microservicios, software distribuido y más.