Introducción a Containers y Docker

La inclusión en contenedores es un enfoque de desarrollo de software en el que una aplicación o un servicio, sus dependencias y su configuración (extraídos como archivos de manifiesto de implementación) se empaquetan como una imagen de contenedor. La aplicación en contenedor puede probarse como una unidad e implementarse como una instancia de imagen de contenedor en el sistema operativo (SO) host.

Del mismo modo que los contenedores de mercancías permiten su transporte por barco, tren o camión independientemente de la carga de su interior, los contenedores de software actúan como una unidad estándar de implementación de software que puede contener diferentes dependencias y código. De esta manera, la inclusión del software en contenedor permite a los desarrolladores y los profesionales de TI implementarlo en entornos con pocas modificaciones o ninguna en absoluto.

Los contenedores también aíslan las aplicaciones entre sí en un sistema operativo compartido. Las aplicaciones en contenedor se ejecutan sobre un host de contenedor que a su vez se ejecuta en el sistema operativo (Linux o Windows). Por lo tanto, los contenedores tienen una superficie significativamente menor que las imágenes de máquina virtual (VM).

Cada contenedor puede ejecutar una aplicación web o un servicio al completo, como se muestra en la figura 2-1. En este ejemplo, el host de Docker es un host de contenedor, y App 1, App 2, Svc 1 y Svc 2 son aplicaciones o servicios en contenedor.

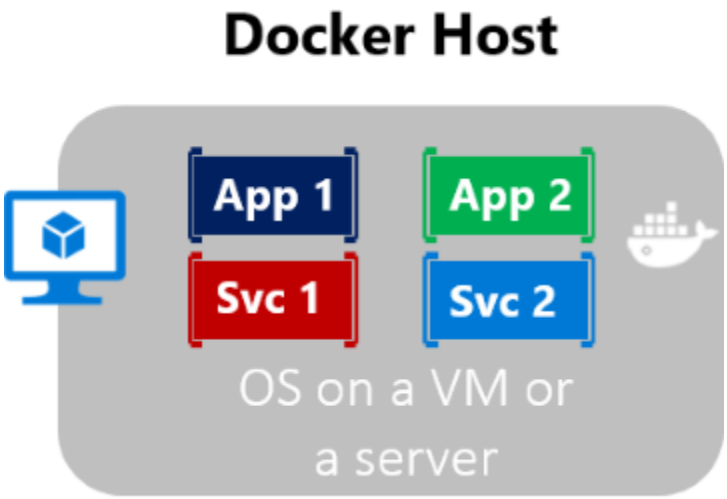


Figura 2-1. Varios contenedores ejecutándose en un host de contenedor.

Otra ventaja de la inclusión en contenedores es la escalabilidad. La creación de contenedores para tareas a corto plazo permite escalar horizontalmente con gran rapidez. Desde el punto de vista de la aplicación, la creación de instancias de una imagen (la creación de un contenedor) es similar a la creación de instancias de un proceso como un servicio o una aplicación web. Pero con fines de confiabilidad, cuando ejecute varias instancias de la misma imagen en varios servidores host, seguramente le interesará que cada contenedor (instancia de imagen) se ejecute en un servidor host o máquina virtual diferente en dominios de error distintos.

En resumen, los contenedores ofrecen las ventajas del aislamiento, la portabilidad, la agilidad, la escalabilidad y el control a lo largo de todo el flujo de trabajo del ciclo de vida de la aplicación. La ventaja más importante es el aislamiento del entorno que se proporciona entre el desarrollo y las operaciones.

¿Qué es Docker?

Docker es un proyecto de código abierto para automatizar la implementación de aplicaciones como contenedores portátiles y autosuficientes que se pueden ejecutar en la nube o localmente. Docker es también una empresa que promueve e impulsa esta tecnología, en colaboración con proveedores de la nube, Linux y Windows.

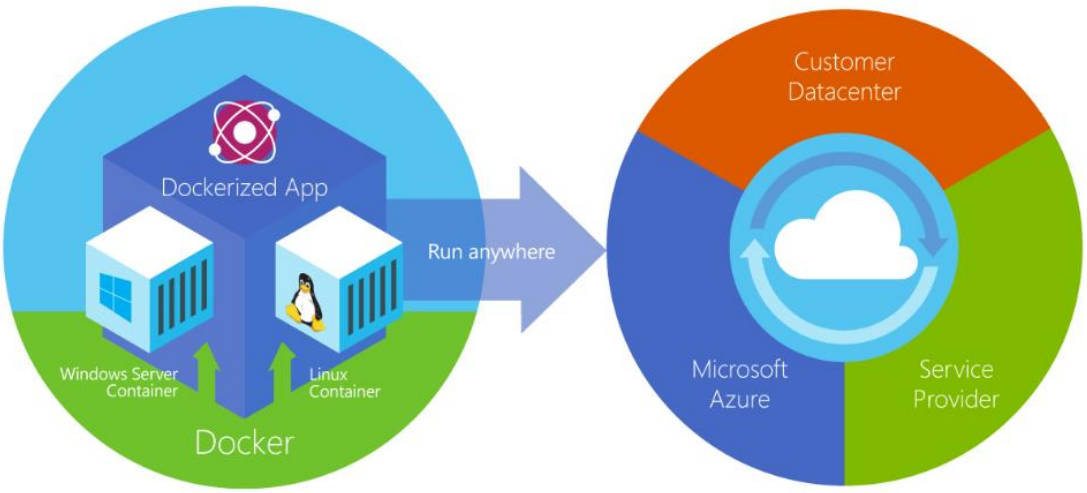


Figura 2-2. Docker implementa contenedores en todas las capas de la nube híbrida.

Los contenedores de Docker se pueden ejecutar en cualquier lugar, a nivel local en el centro de datos de cliente, en un proveedor de servicios externo o en la nube. Los contenedores de imagen de Docker se pueden ejecutar de forma nativa en Linux y Windows. Sin embargo, las imágenes de Windows solo pueden ejecutarse en hosts de Windows y las imágenes de Linux pueden ejecutarse en hosts de Linux y hosts de Windows, donde host significa un servidor o una máquina virtual.

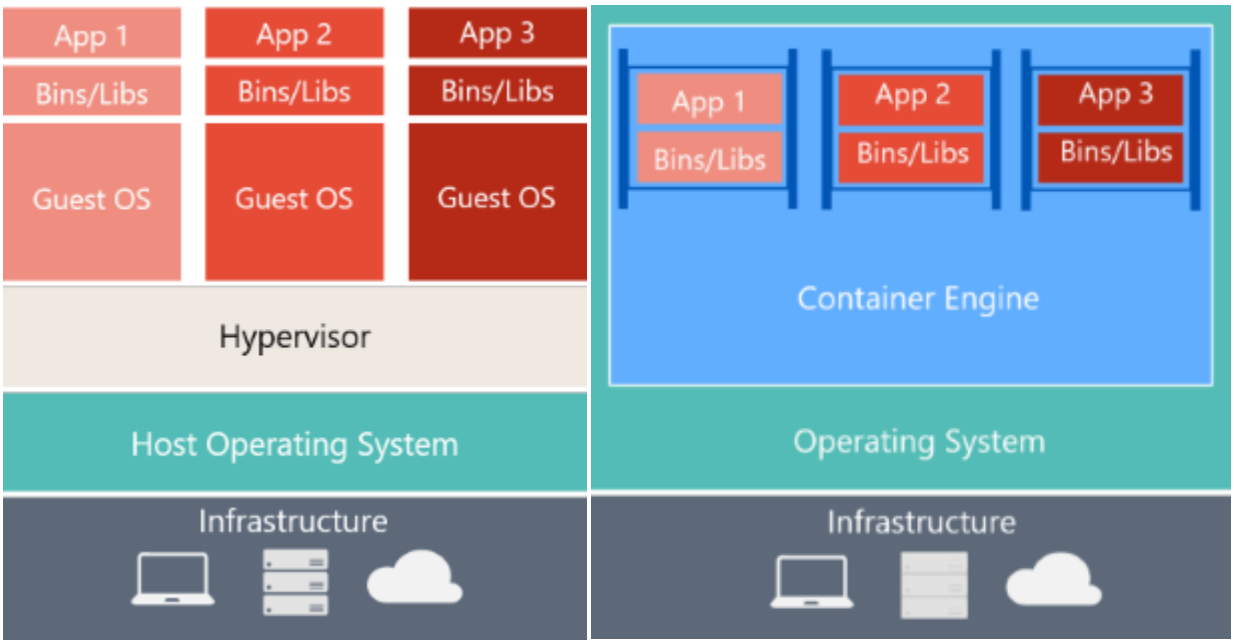
Los desarrolladores pueden usar entornos de desarrollo en Windows, Linux o macOS. En el equipo de desarrollo, el desarrollador ejecuta un host de Docker en que se implementan imágenes de Docker, incluidas la aplicación y sus dependencias. Los desarrolladores que trabajan en Linux o macOS usan un host de Docker basado en Linux y pueden crear imágenes solo para contenedores de Linux. (Los desarrolladores que trabajan en macOS pueden editar código o ejecutar la CLI de Docker en macOS, pero en el momento de redactar este artículo, los contenedores no se ejecutan directamente en macOS). Los desarrolladores que trabajan en Windows pueden crear imágenes para contenedores de Windows o Linux.

Para hospedar contenedores en entornos de desarrollo y proporcionar herramientas de desarrollador adicionales, Docker entrega Docker Community Edition (CE) para Windows o macOS. Estos productos instalan la máquina virtual necesaria (el host de Docker) para hospedar los contenedores. Docker también pone a disposición Docker Enterprise Edition (EE), que está diseñado para el desarrollo empresarial y se usa en los equipos de TI que crean, envían y ejecutan aplicaciones críticas para la empresa en producción.

Comparación de los contenedores de Docker con las máquinas virtuales

En la figura 2-3 se muestra una comparación entre las máquinas virtuales y los contenedores de Docker.

COMPARACIÓN DE LOS CONTENEDORES DE DOCKER CON LAS MÁQUINAS VIRTUALES



Las máquinas virtuales incluyen la aplicación, las bibliotecas o los archivos binarios necesarios y un sistema operativo invitado completo. La virtualización completa requiere más recursos que la inclusión en contenedores.

Los contenedores incluyen la aplicación y todas sus dependencias. Sin embargo, comparten el kernel del sistema operativo con otros contenedores, que se ejecutan como procesos aislados en el espacio de usuario en el sistema operativo host. (Excepto en los contenedores de Hyper-V, en que cada contenedor se ejecuta dentro de una máquina virtual especial por contenedor).

Figura 2-3. Comparación de las máquinas virtuales tradicionales con los contenedores de Docker

Para las máquinas virtuales, hay tres niveles de base en el servidor host, de manera ascendente: infraestructura, sistema operativo host y un hipervisor y, encima de todo eso, cada máquina virtual tiene su propio sistema operativo y todas las bibliotecas necesarias. En el caso de Docker, el servidor host solo tiene la infraestructura y el sistema operativo y, encima de eso, el motor de contenedor, que mantiene el contenedor aislado, pero con el uso compartido de los servicios del sistema operativo de base.

Dado que los contenedores requieren muchos menos recursos (por ejemplo, no necesitan un sistema operativo completo), se inician rápidamente y son fáciles de implementar. Esto permite tener una mayor densidad, lo que significa que se pueden ejecutar más servicios en la misma unidad de hardware, reduciendo así los costos.

Como efecto secundario de la ejecución en el mismo kernel, obtiene menos aislamiento que las máquinas virtuales.

El objetivo principal de una imagen es que hace que el entorno (dependencias) sea el mismo entre las distintas implementaciones. Esto significa que puede depurarlo en su equipo y, a continuación, implementarlo en otra máquina con el mismo entorno garantizado.

Una imagen de contenedor es una manera de empaquetar una aplicación o un servicio e implementarlo de forma confiable y reproducible. Podría decir que Docker no solo es una tecnología, sino también una filosofía y un proceso.

Al usar Docker, no escuchará a los desarrolladores decir "Si funciona en mi máquina, ¿por qué no en producción?". Pueden decir simplemente "Se ejecuta en Docker", porque la aplicación de Docker empaquetada puede ejecutarse en cualquier entorno de Docker compatible, y se ejecuta de la forma prevista en todos los destinos de implementación (como desarrollo, control de calidad, ensayo y producción).

Una analogía simple

Quizás una analogía simple puede ayudar a entender el concepto básico de Docker.

Vamos a remontarnos a la década de 1950 por un momento. No había ningún procesador de texto y las fotocopiadoras se utilizaban en todas partes y de todo tipo.

Imagine que es responsable de enviar lotes de cartas, según proceda, para enviarlas por correo a los clientes en papel y sobres reales, que se entregarán físicamente en la dirección postal de cada cliente (entonces no existía el correo electrónico).

En algún momento, se da cuenta de que las cartas no son más que una composición de un conjunto grande de párrafos, que se eligen y ordenan según proceda, según el propósito de la carga, por lo que diseña un sistema para emitir cartas rápidamente, esperando conseguir una increíble mejora.

El sistema es simple:

1. Comience con un conjunto de hojas transparentes que contienen un párrafo.
2. Para emitir un conjunto de cartas, elija las hojas con los párrafos necesarios, apílelas y alinéelas para que queden y se lean bien.

3. Por último, colóquelas en la fotocopiadora y presione inicio para producir tantas cartas como sean necesarias.

Por tanto, para simplificar, esa es la idea principal de Docker.

En Docker, cada capa es el conjunto resultante de los cambios que se producen en el sistema de archivos después de ejecutar un comando, como instalar un programa.

Por lo tanto, al "Examinar" el sistema de archivos después de que se ha copiado la capa, verá todos los archivos, incluida la capa cuando se instaló el programa.

Puede pensar en una imagen como un disco duro de solo lectura auxiliar listo para instalarse en un "equipo" donde el sistema operativo ya está instalado.

De forma similar, puede pensar en un contenedor como el "equipo" con el disco duro de imagen instalado. El contenedor, como un equipo, se puede apagar o desactivar.

Terminología de Docker

En esta sección se enumeran los términos y las definiciones que debe conocer antes de profundizar en el uso de Docker. Para consultar más definiciones, lea el amplio glosario que Docker proporciona.

Imagen de contenedor: un paquete con todas las dependencias y la información necesarias para crear un contenedor. Una imagen incluye todas las dependencias (por ejemplo, los marcos), así como la configuración de implementación y ejecución que usará el runtime de un contenedor. Normalmente, una imagen se deriva de varias imágenes base que son capas que se apilan unas encima de otras para formar el sistema de archivos del contenedor. Una vez que se crea una imagen, esta es inmutable.

Dockerfile: archivo de texto que contiene instrucciones sobre cómo compilar una imagen de Docker. Es como un script por lotes; la primera línea indica la imagen base con la que se comienza y, después, deben seguirse las instrucciones para instalar programas necesarios, copiar archivos, etc., hasta obtener el entorno de trabajo que se necesita.

Compilación: la acción de crear una imagen de contenedor basada en la información y el contexto que proporciona su Dockerfile, así como archivos adicionales en la carpeta en que se crea la imagen. Puede compilar imágenes con el siguiente comando de Docker:

BashCopiar

```
docker build
```

Contenedor: una instancia de una imagen de Docker. Un contenedor representa la ejecución de una sola aplicación, proceso o servicio. Está formado por el contenido de una imagen de Docker, un entorno de ejecución y un conjunto estándar de instrucciones. Al escalar un servicio, crea varias instancias de un contenedor a partir de la misma imagen. O bien, un proceso por lotes puede crear varios contenedores a partir de la misma imagen y pasar parámetros diferentes a cada instancia.

Volúmenes: ofrece un sistema de archivos grabable que el contenedor puede usar. Puesto que las imágenes son de solo lectura pero la mayoría de los programas necesitan escribir en el sistema de archivos, los volúmenes agregan una capa grabable, encima de la imagen de contenedor, por lo que los programas tienen acceso a un sistema de archivos grabable. El programa no sabe que está accediendo a un sistema de archivos

por capas, que no es más que el sistema de archivos habitual. Los volúmenes residen en el sistema host y los administra Docker.

Etiqueta: una marca o una etiqueta que se puede aplicar a las imágenes para que se puedan identificar diferentes imágenes o versiones de la misma imagen (según el número de versión o el entorno de destino).

Compilación de varias fases: es una característica, desde Docker 17.05 o versiones posteriores, que ayuda a reducir el tamaño de las imágenes finales. En pocas palabras, con la compilación de varias fases se puede usar, por ejemplo, una imagen base grande, que contiene el SDK, para compilar y publicar la aplicación y, después, usar la carpeta de publicación con una imagen base pequeña solo en tiempo de ejecución, para generar una imagen final mucho más pequeña.

Repositorio: una colección de imágenes de Docker relacionadas, etiquetadas con una etiqueta que indica la versión de la imagen. Algunos repositorios contienen varias variantes de una imagen específica, como una imagen que contiene SDK (más pesada), una imagen que solo contiene runtimes (más ligera), etcétera. Estas variantes se pueden marcar con etiquetas. Un solo repositorio puede contener variantes de plataforma, como una imagen de Linux y una imagen de Windows.

Registro: servicio que proporciona acceso a los repositorios. El registro predeterminado para la mayoría de las imágenes públicas es Docker Hub (propiedad de Docker como una organización). Normalmente, un registro contiene repositorios procedentes de varios equipos. Las empresas suelen tener registros privados para almacenar y administrar imágenes que han creado. Azure Container Registry es otro ejemplo.

Imagen multiarquitectura: En el caso de la arquitectura múltiple, es una característica que simplifica la selección de la imagen adecuada, según la plataforma donde se ejecuta Docker. Por ejemplo, si un Dockerfile solicita una imagen base **mcr.microsoft.com/dotnet/sdk:5.0** del Registro, en realidad obtendrá **5.0-nanoserver-1909**, **5.0-nanoserver-1809** o **5.0-buster-slim**, según el sistema operativo en el que se ejecute Docker y la versión.

Docker Hub: registro público para cargar imágenes y trabajar con ellas. Docker Hub proporciona hospedaje de imágenes de Docker, registros públicos o privados, desencadenadores de compilación y enlaces web e integración con GitHub y Bitbucket.

Azure Container Registry: recurso público para trabajar con imágenes de Docker y sus componentes en Azure. Esto proporciona un registro cercano a las implementaciones en Azure que le proporciona control sobre el acceso, lo que le permite usar los grupos y los permisos de Azure Active Directory.

Docker Trusted Registry (DTR) : servicio del registro de Docker (ofrecido por Docker) que se puede instalar de forma local, por lo que se encuentra en el centro de datos y la red de la organización. Es ideal para imágenes privadas que deben administrarse dentro de la empresa. Docker Trusted Registry se incluye como parte del producto Docker Datacenter.

Docker Community Edition (CE) : herramientas de desarrollo para Windows y MacOS para compilar, ejecutar y probar contenedores localmente. Docker CE para Windows proporciona entornos de desarrollo para contenedores de Windows y Linux. El host de Docker de Linux en Windows se basa en una máquina virtual Hyper-V. El host para los contenedores de Windows se basa directamente en Windows. Docker CE para Mac se

basa en el marco del hipervisor de Apple y el hipervisor xhyve, lo que proporciona una máquina virtual de host de Docker de Linux en macOS X. Docker CE para Windows y para Mac sustituye a Docker Toolbox, que se basaba en Oracle VirtualBox.

Docker Enterprise Edition (EE) : versión a escala empresarial de las herramientas de Docker para el desarrollo de Linux y Windows.

Compose: herramienta de línea de comandos y formato de archivo YAML con metadatos para definir y ejecutar aplicaciones de varios contenedores. Primero se define una sola aplicación basada en varias imágenes con uno o más archivos .yml que pueden invalidar los valores según el entorno. Después de crear las definiciones, puede implementar toda la aplicación de varios contenedores con un solo comando (docker-compose up) que crea un contenedor por imagen en el host de Docker.

Clúster: colección de hosts de Docker que se expone como si fuera un solo host de Docker virtual, de manera que la aplicación se puede escalar a varias instancias de los servicios repartidos entre varios hosts del clúster. Los clústeres de Docker se pueden crear con Kubernetes, Azure Service Fabric, Docker Swarm y Mesosphere DC/OS.

Orquestador: herramienta que simplifica la administración de clústeres y hosts de Docker. Los orquestadores permiten administrar las imágenes, los contenedores y los hosts a través de una interfaz de la línea de comandos (CLI) o una interfaz gráfica de usuario. Puede administrar las redes de contenedor, las configuraciones, el equilibrio de carga, la detección de servicios, la alta disponibilidad, la configuración del host de Docker y muchas cosas más. Un orquestador se encarga de ejecutar, distribuir, escalar y reparar las cargas de trabajo a través de una colección de nodos. Normalmente, los

productos del orquestador son los mismos que proporcionan infraestructura de clúster, como Kubernetes y Azure Service Fabric, entre otras ofertas del mercado.

Contenedores, imágenes y registros de Docker

Al usar Docker, un desarrollador crea una aplicación o un servicio y lo empaqueta, junto con sus dependencias, en una imagen de contenedor. Una imagen es una representación estática de la aplicación o el servicio y de su configuración y las dependencias.

Para ejecutar la aplicación o el servicio, se crea una instancia de la imagen de la aplicación para crear un contenedor, que se ejecutará en el host de Docker. Inicialmente, los contenedores se prueban en un entorno de desarrollo o un PC.

Los desarrolladores deben almacenar las imágenes en un registro, que actúa como una biblioteca de imágenes y es necesario cuando se implementa en orquestadores de producción. Docker mantiene un registro público a través de Docker Hub; otros proveedores ofrecen registros para distintas colecciones de imágenes, incluido Azure Container Registry. Como alternativa, las empresas pueden tener un registro privado local para sus propias imágenes de Docker.

En la figura 2-4 se muestra cómo se relacionan las imágenes y los registros de Docker con otros componentes. También se muestran las diversas ofertas de registro de los proveedores.

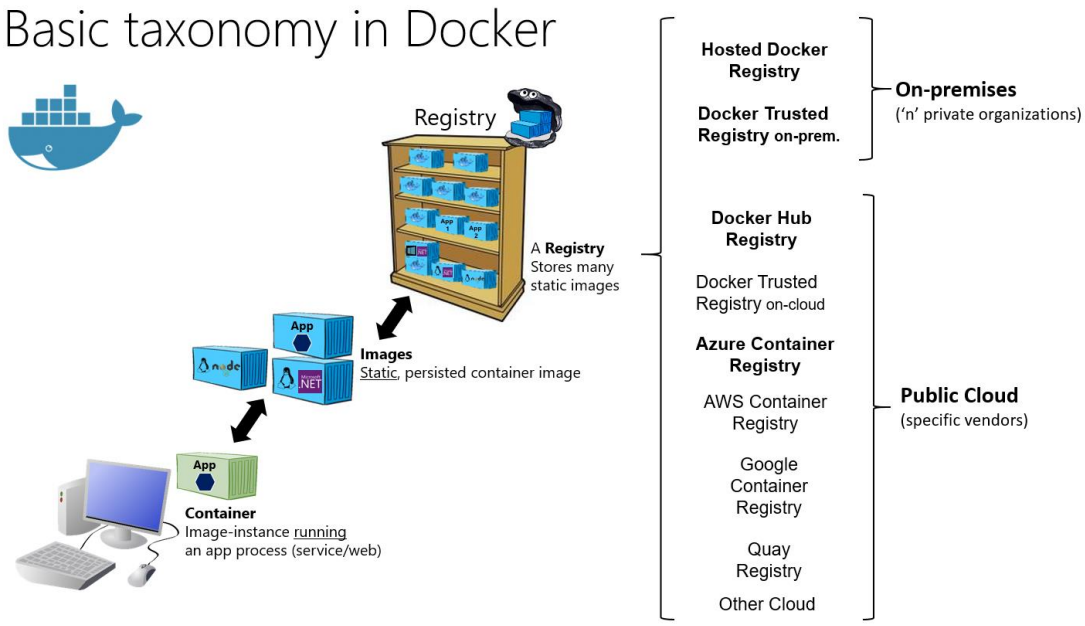


Figura 2-4. Taxonomía de términos de Docker y conceptos

el registro es como una estantería donde las imágenes se almacenan y están disponibles para extraerlas con el fin de compilar contenedores que ejecuten servicios o aplicaciones web. Hay registros de Docker privados a nivel local y en la nube pública. Docker Hub es que un registro público mantenido por Docker; junto con Docker Trusted Registry, una solución a nivel empresarial, Azure ofrece Azure Container Registry. AWS, Google y otros también tienen registros de contenedor.

Colocar imágenes en un registro le permite almacenar fragmentos de la aplicación que son estáticos e inmutables, incluidas todas sus dependencias a nivel de marco. Después, esas imágenes se pueden versionear e implementar en varios entornos y, por tanto, proporcionar una unidad de implementación coherente.

Los registros de imágenes privados ya sean hospedados localmente o en la nube, se recomiendan cuando:

- Las imágenes no deben compartirse públicamente por motivos de confidencialidad.
- Quiere tener una latencia de red mínima entre las imágenes y el entorno de implementación elegido. Por ejemplo, si el entorno de producción es la nube de Azure, probablemente quiera almacenar las imágenes en Azure Container Registry, para que la latencia de red sea mínima. De forma similar, si el entorno de producción es local, puede tener un registro de confianza de Docker local disponible dentro de la misma red local.