Artículo
Ricardo Paiva · 5 sep, 2019 Lectura de 5 min

Contenedores, ¿qué es un contenedor basado en IMÁGENES?

¡Hola a tod@s!

En esta segunda publicación sobre los principios básicos de los contenedores, echaremos un vistazo a los *contenedores basados en imágenes*.

Un **contenedor *basado en imágenes*** es simplemente la representación binaria de un contenedor.

Un contenedor en ejecución o simplemente un *contenedor* es el estado de ejecución relacionado con el contenedor basado en *imágenes*.

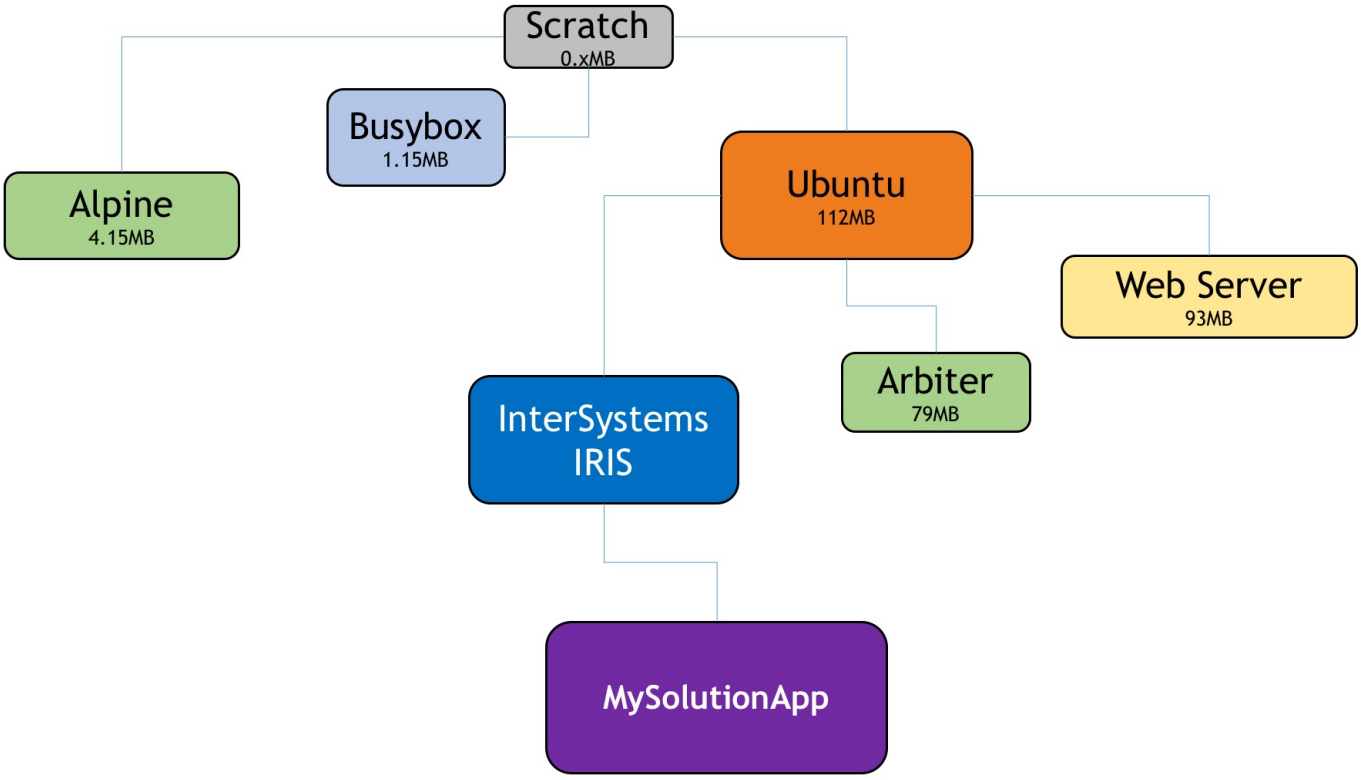
Consulte la primera publicación en la que se explica **qué es un contenedor**.

Los contenedores basados en imágenes consisten en un sustrato básico para el sistema operativo y todo el software que necesitamos para ejecutar nuestro servicio. Aquí utilizo el término *servicio* para una determinada "solución de contenedor" como la descripción general de una solución de software dentro de un contenedor. Hago esto porque podría referirme a un tipo diferente de "soluciones para aplicaciones" o componentes, como un simple servidor web, algo más complicado como un endpoint REST o una solución completamente monolítica encaminada a modernizarse en su SDLC a traves de CI/CD. En una nube nativa, la composición de la arquitectura, que podría incorporar microservicios, el concepto de "servicio" sería lo más cercano a este término.

Podemos pensar en un contenedor basado en imágenes como un paquete estático de tipo VMDK, OVA, OVF, QEMU, AWS AMI, etc. Sin embargo, los contenedores basados en imágenes son diferentes porque cuentan con muchas características interesantes.

Con el contenedor basado en imágenes, tenemos la noción de un padre con su hijo o una imagen derivada de ello. Técnicamente, la noción es similar a la de los snapshots: archivos de bloques que se agregaron a una imagen ya existente. Terminaremos con el concepto de jerarquía en las imágenes, el cual es muy útil. Los contenedores utilizan archivos del sistema de tipo Union-FS que nos ayudan con la jerarquía y la colocación de capas adicionales.

Existen claras ventajas en organizar las imágenes mediante una estructura de árbol (consulte el siguiente diagrama), ya que de este modo todos podemos escoger la fruta que nos gusta :) y por supuesto uno de los resultados es que podemos compartirlas en cada nivel.



Otra forma de pensar en este contenedor y en el proceso de creación de imágenes es su cercanía al uso de un lenguaje orientado a objetos. Esto es el entorno orientado a objetos de los administradores de sistemas, o ingenieros de DevOps como les llamo, ya que heredamos una imagen con todas sus propiedades y más adelante veremos todas sus especificaciones con mayor detalle. También tenemos la opción de anular los componentes heredados (pensar que el archivo de una biblioteca antigua tiene una vulnerabilidad que necesita corregirse). Si representamos esa estructura de árbol con imágenes heredadas o derivadas, sabríamos (y las herramientas del contenedor pueden decírnoslo) en cuál de las imágenes hay un problema o una vulnerabilidad. El problema puede resolverse rápidamente en el nivel particular de esa imagen, y todas las imágenes que dependan de ella podrán entonces heredar el beneficio de esa imagen corregida. En el funcionamiento, la herencia automática se logra simplemente mediante un proceso de reconstrucción. La analogía con la orientación a objetos no termina con las imágenes, sino que se extiende hacia el contenedor en ejecución. Una imagen es como la definición de una clase, mientras que un contenedor (consulte la **publicación anterior)** es la creación de instancias, el objeto, de una imagen.

De todo lo que se expuso anteriormente podemos deducir que otra ventaja de la estructura del árbol es que podemos enfocarnos en cada nivel o imagen para adaptarla.

Capas

Otra característica de los *contenedores basados en imágenes* es que normalmente están formados por varias *capas*.

¿Qué son las capas? Las capas esencialmente son archivos generados por la ejecución de algunos comandos.

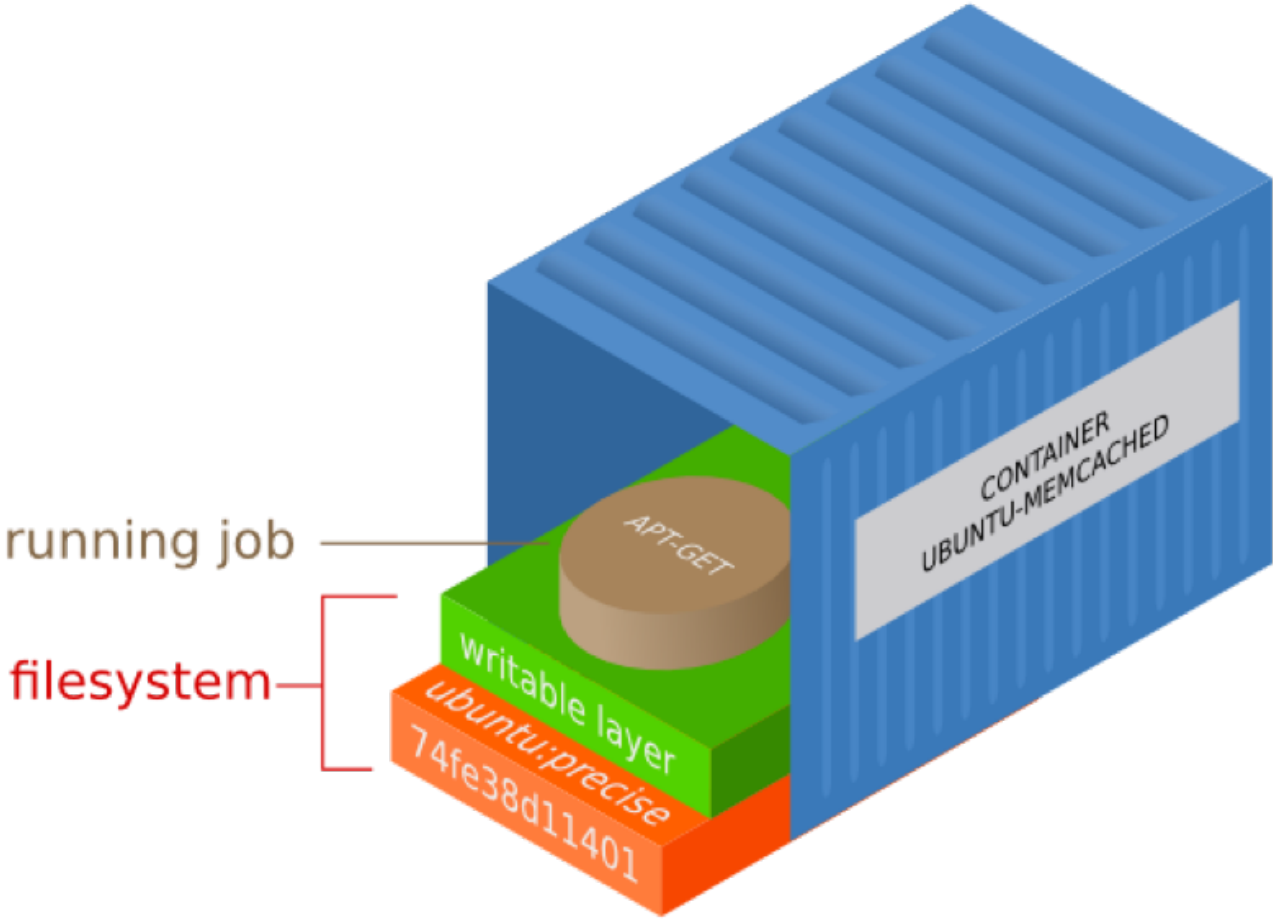
Puede ver todas las capas en el directorio donde Docker guarda sus herramientas, el cual se encuentra debajo del directorio raíz de Docker, que de forma predeterminada es: /var/lib/docker/.

El concepto de las capas es ingenioso porque pueden reutilizarse para varias imágenes, de modo que ahorran espacio de almacenamiento y hacen que la construcción de nuevas imágenes, y el arrastre de estas sea mucho más rápido debido a que se reutilizan en comparación a las que se reconstruyen o extraen de nuevo. Esto también mejora la integración de todas las imágenes.

A las capas también se les llama ***imágenes intermedias***. Cuando todas las imágenes intermedias creadas por el proceso de construcción se especifican juntas mediante el nombre principal y la nomenclatura de etiquetas en uso,

entonces tenemos o estamos tratando con el *contenedor basado en imágenes*.

Cada contenedor que ejecutamos tiene una capa superior de lectura y escritura (consulte la imagen que se muestra a continuación), con una serie de capas de solo lectura debajo de ella. Todas esas capas de solo lectura son el resultado de los comandos que se ejecutaron cuando se construyó el contenedor.



La razón por la que los contenedores se llaman efímeros es porque normalmente, cuando detenemos un contenedor, lo desechamos y esa capa superior de lectura/escritura se pierde para siempre. Sin embargo, antes de eliminar el contenedor puede aceptar sus cambios. La operación de aceptación guardaría esa capa superior de lectura/escritura en una capa de solo lectura para la siguiente ejecución del contenedor. Efectivamente, ahora tiene un contenedor nuevo y normalmente se le pedirá que lo guarde con un nombre y una etiqueta nueva.

En resumen, Docker es el motor de contenedores más utilizado y la empresa que nos permitió a todos utilizar fácilmente los contenedores agregó una imagen muy intuitiva sobre cómo se realiza todo el proceso de las API con un kernel de bajo nivel, el cual además se ocupa de *empaquetar* la solución. Este es un paquete especial (como el de la *imagen*), ya que se trata de un *ejecutable*. Docker ofrece una interacción natural con el Docker Engine (dockerd) mediante una sencilla API REST y una Interfaz de Línea de Comandos (CLI) muy intuitiva para el cliente, la cual nos permite lidiar con todo lo que esté relacionado con los contenedores.

Docker nos ayuda con lo siguiente:

- Para ejecutarlos
  - $ docker run
- Para guardarlos, almacenarlos y recuperarlos
  - $ docker commit ¦ push ¦ pull
- Para crearlos o construirlos
  - $ docker build

y hace más valiosa toda la "experiencia con el contenedor".

En la siguiente publicación, hablaré del proceso de construcción.

Mientras tanto, disfrute de explorar imágenes predefinidas mediante **Docker Hub**.

#Contenedorización #Despliegue #DevOps #Docker #Nube #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/contenedores-%C2%BFqu%C3%A9-es-un-contenedor-basado-en-im%C3%A1genes