Administrar datos en contenedores

Tiempo estimado de lectura: 14 minutos

En esta sección aprenderá cómo administrar datos dentro y entre los contenedores de Docker.

Vas a ver las dos maneras principales de administrar datos con Docker Engine.

- Volúmenes de datos
- Contenedores de volumen de datos

Volúmenes de datos

Un *volumen de datos* es un directorio especialmente designado dentro de uno o más contenedores que elimina el *sistema de archivos de Unión* (https://docs.docker.com/glossary/?term=Union%20file%20system) . Los volúmenes de datos proporcionan varias características útiles para datos persistentes o compartidos:

- Los volúmenes se inicializan cuando se crea un contenedor. Si la imagen principal del contenedor contiene datos en el punto de montaje especificado, los datos existentes se copian en el nuevo volumen al inicializarse el volumen. (Tenga en cuenta que esto no se aplica al montar un directorio de host (/engine/tutorials/dockervolumes/#mount-a-host-file-as-a-data-volume) .)
- Los volúmenes de datos pueden ser compartidos y reutilizados entre contenedores.
- Los cambios en un volumen de datos se realizan directamente.
- Los cambios en un volumen de datos no se incluirán cuando actualice una imagen.
- Los volúmenes de datos persisten incluso si se elimina el contenedor.

Los volúmenes de datos están diseñados para persistir los datos, independientemente del ciclo de vida del contenedor. Por tanto, Docker *nunca* borra automáticamente los volúmenes cuando elimina un contenedor ni recoge basura los volúmenes que ya no son referenciados por un contenedor.

Añadir un volumen de datos

Puede agregar un volumen de datos a un contenedor utilizando el `-v` indicador con el comando `docker create` y `docker run` . Puede utilizar `-v` varias veces para montar varios volúmenes de datos. Ahora, monte un solo volumen en su contenedor de aplicaciones web.

```
$ docker run -d -P --name web -v /webapp training/webapp python app.py
```

Esto creará un nuevo volumen dentro de un contenedor en `/webapp` .

> **Nota** : También puede utilizar la `VOLUME` instrucción en a `Dockerfile` para agregar uno o más volúmenes nuevos a cualquier contenedor creado a partir de esa imagen.

Localice un volumen

Puede localizar el volumen en el host utilizando el `docker inspect` comando.

```
$ docker inspect web
```

La salida proporcionará detalles sobre las configuraciones del contenedor incluyendo los volúmenes. La salida debe ser similar a lo siguiente:

```
...
"Mounts": [
    {
        "Name": "fac362...80535",
        "Source": "/var/lib/docker/volumes/fac362...80535/_data",
        "Destination": "/webapp",
        "Driver": "local",
        "Mode": "",
        "RW": true,
        "Propagation": ""
    }
]
...
```

Usted notará en el anterior `Source` está especificando la ubicación en el host y `Destination` está especificando la ubicación del volumen dentro del contenedor. `RW` Muestra si el volumen es de lectura / escritura.

Monte un directorio de host como un volumen de datos

Además de crear un volumen utilizando la `-v` bandera, también puede montar un directorio desde el host del motor Docker en un contenedor.

```
$ docker run -d -P --name web -v /src/webapp:/webapp training/webapp python app.py
```

Este comando monta el directorio de host,, `/src/webapp` en el contenedor en `/webapp` . Si la ruta `/webapp` ya existe dentro de la imagen del contenedor, el `/src/webapp` montaje se superpone pero no elimina el contenido preexistente. Una vez que el soporte se retira, el contenido es accesible de nuevo. Esto es coherente con el comportamiento esperado del `mount` comando.

El `container-dir` debe siempre ser un camino absoluto como `/src/docs` . El `host-dir` puede ser un camino absoluto como `/dst/docs` en Linux o `C:\dst\docs` en Windows, o un `name` valor. Si proporciona una ruta absoluta para el `host-dir` , Docker bind-monta a la ruta que especifique. Si usted proporciona a `name` , Docker crea un volumen nombrado por eso `name` .

Un `name` valor debe comenzar con un carácter alfanumérico, seguido por `a-z0-9` , `_` (subrayado), `.` (punto) o `-` (guión). Una ruta absoluta comienza con una `/` barra (barra diagonal).

Por ejemplo, puede especificar uno `/foo` o `foo` para un `host-dir` valor. Si suministra el `/foo` valor, el motor Docker crea un montaje de enlace. Si suministra la `foo` especificación, Docker Engine crea un volumen con nombre.

Si utiliza Docker Machine en Mac o Windows, su daemon Docker Engine sólo tiene acceso limitado a su sistema de archivos macOS o Windows. Docker Machine intenta compartir automáticamente su directorio `/Users` (macOS) o `C:\Users` (Windows). Por lo tanto, puede montar archivos o directorios en macOS utilizando.

```
docker run -v /Users/<path>:/<container path> ...
```

En Windows, monte directorios utilizando:

```
docker run -v c:\<path>:c:\<container path>
```

Todos los demás caminos vienen del sistema de archivos de su máquina virtual, por lo que si desea que otra carpeta de host esté disponible para compartir, debe realizar un trabajo adicional. En el caso de VirtualBox es necesario que la carpeta host esté disponible como una carpeta compartida en VirtualBox. A continuación, puede montarlo utilizando el `-v` indicador Docker .

El montaje de un directorio de host puede ser útil para las pruebas. Por ejemplo, puede montar código fuente dentro de un contenedor. A continuación, cambie el código fuente y vea su efecto en la aplicación en tiempo real. El directorio en el host debe ser especificado como una ruta absoluta y si el directorio no existe, el daemon de Docker Engine lo crea automáticamente para usted.

Docker volúmenes por defecto para montar en modo de lectura-escritura, pero también puede configurarlo para ser montado de sólo lectura.

```
$ docker run -d -P --name web -v /src/webapp:/webapp:ro training/webapp python app.py
```

Aquí montó el mismo `/src/webapp` directorio pero ha agregado la `ro` opción de especificar que el montaje debe ser de sólo lectura.

También puede relajar los requisitos de coherencia de un directorio montado para mejorar el rendimiento añadiendo la `cached` opción:

```
$ docker run -d -P --name web -v /src/webapp:/webapp:cached training/webapp python app.py
```

La `cached` opción típicamente mejora el rendimiento de cargas de trabajo de lectura pesada en Docker para Mac, a costa de alguna incoherencia temporal entre el host y el contenedor. En otras plataformas, `cached` actualmente no tiene efecto. El artículo Caching guiado por el usuario en Docker para Mac (https://blog.docker.com/2017/05/user-guided-caching-in-docker-for-mac/) da más detalles sobre el comportamiento de `cached` en macOS.

> **Nota** : El directorio del host es, por su naturaleza, dependiente del host. Por esta razón, no se puede montar un directorio de host desde `Dockerfile` , la `VOLUME` instrucción no es compatible con pasar a `host-dir` , porque las imágenes construidas deben ser portátiles. Un directorio de host no estaría disponible en todos los hosts potenciales.

Monte un volumen de almacenamiento compartido como un volumen de datos

Además de montar un directorio de host en su contenedor, algunos complementos de volumen de (https://docs.docker.com/engine/extend/plugins_volume/) Docker le permiten provisionar y montar almacenamiento compartido, como iSCSI, NFS o FC.

Una ventaja de usar volúmenes compartidos es que son independientes del host. Esto significa que un volumen puede estar disponible en cualquier host en el que se inicie un contenedor, siempre y cuando tenga acceso al backend de almacenamiento compartido y tenga instalado el complemento.

Una forma de utilizar controladores de volumen es a través del `docker run` comando. Los controladores de volumen crean volúmenes por nombre, en lugar de por ruta como en los otros ejemplos.

El comando siguiente crea un volumen denominado, llamado `my-named-volume` , utilizando el `convoy` controlador de volumen (`convoy` es un complemento para una variedad de back-end de almacenamiento) y lo hace disponible dentro del contenedor en `/webapp` . Antes de ejecutar el comando, instale y configure el convoy (https://github.com/rancher/convoy#quick-start-guide) . Si no desea instalar `convoy` , sustituir `convoy` con `local` en el ejemplo de los comandos a continuación para utilizar el `local` controlador.

```
$ docker run -d -P \
  --volume-driver=convoy \
  -v my-named-volume:/webapp \
  --name web training/webapp python app.py
```

También puede utilizar el `docker volume create` comando, para crear un volumen antes de usarlo en un contenedor.

El ejemplo siguiente también crea el `my-named-volume` volumen, esta vez utilizando el `docker volume create` comando. Las opciones se especifican como pares clave-valor en el formato `o=<key>=<value>` .

```
$ docker volume create -d convoy --opt o=size=20GB my-named-volume

$ docker run -d -P \
  -v my-named-volume:/webapp \
  --name web training/webapp python app.py
```

Una lista de complementos disponibles, incluidos los complementos de volumen, está disponible aquí (https://docs.docker.com/engine/extend/legacy_plugins/) .

Etiquetas de volumen

Los sistemas de etiquetado como SELinux requieren que las etiquetas adecuadas se colocan en el contenido de volumen montado en un contenedor. Sin una etiqueta, el sistema de seguridad puede impedir que los procesos que se ejecutan dentro del contenedor utilicen el contenido. De forma predeterminada, Docker no cambia las etiquetas establecidas por el sistema operativo.

Para cambiar una etiqueta en el contexto del contenedor, puede agregar cualquiera de los dos sufijos `:z` o `:Z` al montaje de volumen. Estos sufijos le indican a Docker que vuelva a etiquetar los objetos de archivo en los volúmenes compartidos. La `z` opción le dice a Docker que dos contenedores comparten el contenido del volumen. Como resultado, Docker etiqueta el contenido con una etiqueta de contenido compartido. Las etiquetas de volumen compartido permiten a todos los contenedores leer y escribir contenido. La `Z` opción le dice a Docker que etiquete el contenido con una etiqueta privada sin compartir. Sólo el contenedor actual puede utilizar un volumen privado.

Monta un archivo host como un volumen de datos

La `-v` bandera también se puede utilizar para montar un solo archivo - en lugar de *sólo* directorios - desde la máquina host.

```
$ docker run --rm -it -v ~/.bash_history:/root/.bash_history ubuntu /bin/bash
```

Esto lo dejará en un shell de bash en un nuevo contenedor, tendrá su historial de bash desde el host y al salir del contenedor, el host tendrá el historial de los comandos escritos mientras está en el contenedor.

> **Nota** : Muchas herramientas se utilizan para editar archivos incluyendo `vi` y `sed --in-place` pueden resultar en un cambio de inodo. Desde Docker v1.1.0, esto producirá un error como " *sed: no se puede renombrar ./sedKdJ9Dy: Dispositivo o recurso ocupado* ". En el caso de que desee editar el archivo montado, a menudo es más fácil montar el directorio padre.

Creación y montaje de un contenedor de volumen de datos

Si usted tiene algunos datos persistentes que desea compartir entre los contenedores, o si desea utilizar desde recipientes no persistentes, lo mejor es crear un recipiente de volumen de datos con nombre y, a continuación, montar los datos de ella.

Vamos a crear un nuevo contenedor con nombre con un volumen para compartir. Mientras este contenedor no ejecuta una aplicación, reutiliza la `training/postgres` imagen para que todos los contenedores estén utilizando capas en común, lo que ahorra espacio en disco.

```
$ docker create -v /dbdata --name dbstore training/postgres /bin/true
```

A continuación, puede utilizar el `--volumes-from` indicador para montar el `/dbdata` volumen en otro contenedor.

```
$ docker run -d --volumes-from dbstore --name db1 training/postgres
```

Y otro:

```
$ docker run -d --volumes-from dbstore --name db2 training/postgres
```

En este caso, si la `postgres` imagen contenía un directorio llamado y `/dbdata` luego montar los volúmenes del `dbstore` contenedor oculta los `/dbdata` archivos de la `postgres` imagen. El resultado es que sólo los archivos del `dbstore` contenedor son visibles.

Puede utilizar varios `--volumes-from` parámetros para combinar volúmenes de datos de varios contenedores. Para obtener información detallada acerca de cómo `--volumes-from` ver los Volúmenes de montaje de contenedor (https://docs.docker.com/engine/reference/commandline/run/#mount-volumes-from-container-volumes-from) en la `run` referencia de comando.

También puede extender la cadena montando el volumen que provenía del `dbstore` contenedor en otro contenedor a través de los contenedores `db1` o `db2` .

```
$ docker run -d --name db3 --volumes-from db1 training/postgres
```

Si retira los contenedores que se montan los volúmenes, entre ellos el primer `dbstore` recipiente o los recipientes posteriores `db1` y `db2` , no se eliminarán los volúmenes. Para eliminar el volumen del disco, debe hacer una llamada explícita `docker rm -v` contra el último contenedor con una referencia al volumen. Esto le permite actualizar o migrar eficazmente volúmenes de datos entre contenedores.

> **Nota** : Docker no le avisará cuando retire un contenedor *sin* proporcionar la `-v` opción de eliminar sus volúmenes. Si elimina contenedores sin utilizar la `-v` opción, puede terminar con volúmenes "colgantes"; Volúmenes que ya no son referenciados por un contenedor. Puede utilizar `docker volume ls -f dangling=true` para encontrar volúmenes colgantes y utilizar `docker volume rm <volume name>` para quitar un volumen que ya no es necesario.

Copia de seguridad, restauración o migración de volúmenes de datos

Otra función útil que podemos realizar con volúmenes es utilizarlos para copias de seguridad, restauraciones o migraciones. Para ello, utilice el `--volumes-from` indicador para crear un nuevo contenedor que monte ese volumen, así:

```
$ docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /dbdata
```

Aquí has lanzado un nuevo contenedor y montado el volumen desde el `dbstore` contenedor. A continuación, ha montado un directorio de host local como `/backup` . Por último, que haya pasado un comando que utiliza `tar` una copia de seguridad del contenido del `dbdata` volumen a un `backup.tar` archivo dentro de nuestro `/backup` directorio. Cuando el comando se completa y se detiene el contenedor nos vamos a quedar con una copia de seguridad de nuestro `dbdata` volumen.

A continuación, puede restaurarlo en el mismo contenedor u otro que haya realizado en otro lugar. Cree un contenedor nuevo.

```
$ docker run -v /dbdata --name dbstore2 ubuntu /bin/bash
```

A continuación, desactive el archivo de copia de seguridad en el volumen de datos del nuevo contenedor.

```
$ docker run --rm --volumes-from dbstore2 -v $(pwd):/backup ubuntu bash -c "cd /dbdata && tar xvf /backup/backup.tar --strip 1"
```

Puede utilizar las técnicas anteriores para automatizar la copia de seguridad, la migración y restaurar las pruebas utilizando sus herramientas preferidas.

Enumerar todos los volúmenes

Puede listar todos los volúmenes existentes utilizando `docker volume ls` .

```
$ docker volume ls
DRIVER              VOLUME NAME
local               ec75c47aa8b8c61fdabcf37f89dad44266841b99dc4b48261a4757e70357ec06
local               f73e499de345187639cdf3c865d97f241216c2382fe5fa67555c64f258892128
local               tmp_data
```

Eliminar volúmenes

Un volumen de datos Docker persiste después de que se elimina un contenedor. Puede crear volúmenes con nombre o anónimo. Los volúmenes nombrados tienen una forma de fuente específica fuera del contenedor, por ejemplo `awesome:/bar` . Los volúmenes anónimos no tienen ninguna fuente específica. Cuando se elimina el contenedor, debe instruir al daemon Docker Engine para que limpie los volúmenes anónimos. Para ello, utilice la `--rm` opción, por ejemplo:

```
$ docker run --rm -v /foo -v awesome:/bar busybox top
```

Este comando crea un `/foo` volumen anónimo . Cuando se retira el contenedor, el Docker Engine elimina el `/foo` volumen pero no el `awesome` volumen.

Para eliminar todos los volúmenes no utilizados y liberar espacio,

```
$ docker volume prune
```

Se eliminarán todos los volúmenes no utilizados que no están asociados con ningún contenedor.

Consejos importantes sobre el uso de volúmenes compartidos

Múltiples contenedores también pueden compartir uno o más volúmenes de datos. Sin embargo, varios contenedores que escriben en un único volumen compartido pueden provocar daños en los datos. Asegúrese de que sus aplicaciones están diseñadas para escribir en almacenes de datos compartidos.

Los volúmenes de datos son directamente accesibles desde el host Docker. Esto significa que puede leer y escribir con las herramientas normales de Linux. En la mayoría de los casos no debe hacer esto, ya que puede causar daños en los datos si sus contenedores y aplicaciones no son conscientes de su acceso directo.

Ejemplos (https://docs.docker.com/glossary/?term=Examples) , Uso (https://docs.docker.com/glossary/?term=Usage) , volumen (https://docs.docker.com/glossary/?term=volume) , docker (https://docs.docker.com/glossary/?term=docker) , documentación (https://docs.docker.com/glossary/?term=documentation) , guía del usuario (https://docs.docker.com/glossary/?term=user%20guide) , datos (https://docs.docker.com/glossary/?term=data) , volúmenes (https://docs.docker.com/glossary/?term=volumes)

¿Qué es Docker? (https://www.docker.com/what-docker)

¿Qué es un contenedor? (https://www.docker.com/what-container)

Casos de Uso (https://www.docker.com/use-cases)

Clientes (https://www.docker.com/customers)

Fogonadura (https://www.docker.com/partners/partner-program)

Para el gobierno (https://www.docker.com/industry-government)

Acerca de Docker (https://www.docker.com/company)

administración (https://www.docker.com/company/management)

Prensa y Noticias (https://www.docker.com/company/news-and-press)

Empleo (https://www.docker.com/careers)

Producto (https://www.docker.com/products/overview)

Precio (https://www.docker.com/pricing)

Edición de comunidad (https://www.docker.com/docker-community)

Edición de Empresa (https://www.docker.com/enterprise)

Docker Datacenter (https://www.docker.com/products/docker-datacenter)

Docker Cloud (https://cloud.docker.com/)

Tienda Docker (https://store.docker.com/)

Docker para Mac (https://www.docker.com/docker-mac)

Docker para Windows (PC) (https://www.docker.com/docker-windows)

Docker para AWS (https://www.docker.com/docker-aws)

Docker para Azure (https://www.docker.com/docker-microsoft-azure)

Docker para Windows Server (https://www.docker.com/docker-windows-server)

Docker para distribución CentOS (https://www.docker.com/docker-centos)

Docker para Debian (https://www.docker.com/docker-debian)

Docker para Fedora® (https://www.docker.com/docker-fedora)

Docker para Oracle Enterprise Linux (https://www.docker.com/docker-oracle-linux)

Docker para RHEL (https://www.docker.com/docker-rhel)

Docker para SLES (https://www.docker.com/docker-sles)

Docker para Ubuntu (https://www.docker.com//docker-ubuntu)

Documentación (/)

Aprender (https://www.docker.com/docker)

Blog (https://blog.docker.com)

Formación (https://training.docker.com/)

Apoyo (https://www.docker.com/docker-support-services)

Base de conocimientos (https://success.docker.com/kbase)

Recursos (https://www.docker.com/products/resources)

Comunidad (https://www.docker.com/docker-community)

Fuente abierta (https://www.docker.com/technologies/overview)

Eventos (https://www.docker.com/community/events)

Foros (https://forums.docker.com/)

Capitanes del muelle (https://www.docker.com/community/docker-captains)

Becas (https://www.docker.com/docker-community/scholarships)

Noticias de la comunidad (https://blog.docker.com/curated/)

Estado (http://status.docker.com/) Seguridad (https://www.docker.com/docker-security) Legal (https://www.docker.com/legal)

Contacto (https://www.docker.com/company/contact)