MANUAL BÁSICO DOCKER

INDICE DE CONTENIDOS

Requisitos previos

Necesitará lo siguiente para seguir este tutorial:

- Un sistema operativo basado en Ubuntu
- Una cuenta en Docker Hub, si desea crear sus propias imágenes y hacer el push a Docker Hub, tal como se indica en el paso 7 y 8.

Paso 1 — Instalar Docker

Es posible que el paquete de instalación de Docker que está disponible en el repositorio oficial de Ubuntu no sea la última versión. Vamos a instalar Docker desde el repositorio oficial de Docker para asegurarnos de tener la última versión. Para hacer esto, vamos a agregar una nueva fuente de paquete, la clave GPG de Docker para asegurar que las descargas sean válidas y después vamos a instalar el paquete.

Primero, actualice su lista de paquetes existente:

$ sudo apt update

A continuación, instale algunos paquetes de requisitos previos que le permiten a apt usar paquetes mediante HTTPS:

$ sudo apt install apt-transport-https ca-certificates curl software-properties-common

Luego, agregue la clave GPG para el repositorio oficial de Docker a su sistema:

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Agregue el repositorio de Docker a las fuentes de APT:

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

Posteriormente, actualice la base de datos de paquetes usando los paquetes de Docker del repositorio que acaba de agregar:

$ sudo apt update

Asegúrese de que va a instalar desde el repositorio de Docker en vez del repositorio de Ubuntu predeterminado:

$ apt-cache policy docker-ce

Por último, instalamos Docker:

$ sudo apt install docker-ce

Ahora debería tener Docker instalado, el daemon iniciado, y el proceso habilitado para iniciar durante el arranque. Verifique que se esté ejecutando:

$ sudo systemctl status docker

Instalar Docker ahora no solamente le ofrece el servicio Docker (daemon), sino también la utilidad de línea de comandos docker o el cliente Docker. Más adelante en este tutorial, vamos a explorar cómo usar el comando docker.

Paso 2 — Ejecutar el comando Docker sin sudo (Opcional)

De forma predeterminada, el comando docker solamente puede ejecutarse por el usuario de **root** o por un usuario en el grupo **docker**, el cual se crea automáticamente durante la instalación de Docker. Si intenta ejecutar el comando docker sin prefijarlo con sudo o sin estar en el grupo **docker**, el resultado será como el siguiente:

Output
docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.
See 'docker run --help'.

Agregue su nombre de usuario al grupo docker si quiere evitar escribir sudo siempre que deba ejecutar el comando docker:

sudo usermod -aG docker ${USER}

Para aplicar la nueva membresía de grupo, debe cerrar sesión en el servidor y volver a iniciarla, o puede escribir lo siguiente:

su - ${USER}

Se le pedirá que ingrese la contraseña de su usuario para poder continuar.

Confirme que se haya agregado su usuario al grupo de **docker** escribiendo:

id -nG

Si necesita agregar un usuario al grupo de docker y no ha iniciado sesión como ese usuario, declare tal nombre de usuario explícitamente usando:

sudo usermod -aG docker username

Para el resto de este artículo, se asume que está ejecutando el comando de docker como un usuario que es parte del grupo de **docker**. Si opta por no hacerlo, anteponga los comandos con sudo.

A continuación, vamos a explorar el comando docker.

Paso 3 — Usar el comando Docker

Usar docker consiste en pasarle una cadena de opciones y comandos seguidos de argumentos. La sintaxis sería la siguiente:

docker [option] [command] [arguments]

Para ver todos los subcomandos disponibles, ingrese:

docker

Si desea ver las opciones disponibles para un comando específico, ingrese:

docker docker-subcommand --help

Si desea ver la información sobre Docker de todo el sistema, use:

docker info

Vamos a explorar algunos de estos comandos. Vamos a empezar trabajando con imágenes.

Paso 4 — Trabajo con imágenes de Docker

Los contenedores Docker se forman a partir de imágenes de Docker. De forma predeterminada, Docker extrae estas imágenes de Docker Hub, un registro de Docker administrado por Docker, la empresa responsable del proyecto Docker. Cualquier persona es capaz de alojar sus imágenes Docker en Docker Hub, por lo tanto, la mayoría de las aplicaciones y distribuciones de Linux que necesitará tendrán las imágenes alojadas ahí mismo.

Para verificar si puede acceder y descargar imágenes desde Docker Hub, ingrese:

docker run hello-world

Puede buscar imágenes disponibles en Docker Hub usando el comando docker con el subcomando de search. Por ejemplo, para buscar la imagen de Ubuntu, ingrese:

docker search ubuntu

En la columna nombrada **OFICIAL**, **OK** indica una imagen que fue creada y soportada por la empresa que respalda el proyecto. Una vez que haya identificado la imagen que quiera usar, puede descargarla a su computadora mediante el subcomando de pull.

Para descargar la imagen de ubuntu oficial a su computadora, ejecute el siguiente comando:

docker pull ubuntu

Tras descargar una imagen, puede ejecutar un contenedor usando la imagen descargada con el subcomando de run. Como vio con el ejemplo de hello-world, si no se ha descargado una imagen al ejecutar docker con el subcomando de run, el cliente Docker primero descargará la imagen y luego ejecutará un contenedor usando la misma.

Para ver las imágenes que se descargaron a su computadora, ingrese:

docker images

El resultado debería parecerse a esto:

Output

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	113a43faa138	4 weeks ago	81.2MB
hello-world	latest	e38bc07ac18e	2 months ago	1.85kB

Como verá más adelante en este tutorial, pueden modificarse y usarse las imágenes que use para ejecutar contenedores para generar imágenes nuevas, las que después pueden cargarse (el término técnico es *pushed*) a Docker Hub u otros registros de Docker.

Vamos a ver más detalladamente cómo ejecutar contenedores.

Paso 5 — Ejecutar un contenedor Docker

El contenedor hello-world que ejecutó durante el paso anterior es un ejemplo de un contenedor que se ejecuta y se va tras emitir un mensaje de prueba. Los contenedores pueden ser mucho más útiles que eso, y pueden ser interactivos. Después de todo, se parecen a máquinas virtuales, nada más que tiene más recursos.

Para dar un ejemplo, ejecutemos un contenedor utilizando la última imagen de Ubuntu. La combinación de los switch **-i** y **-t** le ofrece acceso interactivo a shell en el contenedor:

docker run -it ubuntu

Su línea de comandos debería cambiar para reflejar el hecho de que ahora está trabajando dentro del contenedor y debería verse de esta manera:

root@d9b100f2f636:/#

Note la identificación del contenedor en la línea de comandos. En este ejemplo, es d9b100f2f636. Va a requerir esa identificación de contenedor más adelante para identificar el contenedor cuando quiera eliminarlo.

Ahora puede ejecutar cualquier comando dentro del contenedor. Por ejemplo, vamos a actualizar la base de datos del paquete dentro del contenedor. No es necesario que prefije algún comando con sudo porque está trabajando dentro del contenedor como el usuario de **root**:

apt update

Luego, instale cualquier aplicación en él. Vamos a instalar Node.js:

apt install nodejs

Esto instala Node.js en el contenedor desde el repositorio oficial de Ubuntu. Una vez que termine la instalación, verifique que Node.js esté instalado:

node -v

Verá que el número de versión se muestra en su terminal:

v8.10.0

Los cambios que haga dentro del contenedor únicamente se aplicarán a tal contenedor.

Si desea salir del contenedor, ingrese exit en la línea.

A continuación, vamos a ver cómo gestionar los contenedores en nuestro sistema.

Paso 6 — Gestionar los contenedores de Docker

Una vez que haya estado usando Docker por un tiempo, tendrá varios contenedores activos (siendo ejecutados) e inactivos en su computadora. Si desea ver los que **están activos**, use:

docker ps

Verá un resultado parecido al de abajo:

CONTAINER ID IMAGE COMMAND CREATED

En este tutorial, comenzó teniendo dos contenedores: uno de la imagen de hello-world y otro de la imagen de ubuntu. Ninguno de los contenedores se sigue ejecutando, pero siguen existiendo en su sistema.

Para ver todos los contenedores, tanto los activos como los inactivos, ejecute docker ps con el switch -a:

docker ps -a

Verá un resultado parecido a este:

d9b100f2f636 ubuntu "/bin/bash" About an hour ago Exited (0) 8 minutes ago sharp_volhard

01c950718166 hello-world "/hello" About an hour ago Exited (0) About an hour ago festive_williams

Si desea ver el último contenedor que creó, páselo al switch -l:

docker ps -l

```
chema@X510:~$ docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
3e3c8e7e9cc7        tomsik68/xampp      "sh /startup.sh"    52 minutes ago      Exited (0) 51 minutes ago                       stupefied_wilson
```

Para iniciar un contenedor que se haya detenido, use docker start, seguido de la identificación o el nombre del contenedor. Vamos a empezar con el contenedor basado en Ubuntu cuya identificación era d9b100f2f636:

docker start d9b100f2f636

Se iniciará el contenedor, y puede usar docker ps para ver su estado:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d9b100f2f636 ubuntu "/bin/bash" About an hour ago Up 8 seconds sharp_volhard

Para detener un contenedor que se está ejecutando, use la función de docker stop, seguido de la identificación o el nombre del contenedor. Esta vez, vamos a usar el nombre que Docker le asignó al contenedor, que es sharp_volhard:

docker stop sharp_volhard

Una vez que decida que ya no necesita un contenedor, puede eliminarlo usando el comando docker rm, otra vez usando la identificación o el nombre del contenedor. Use el comando docker ps -a para encontrar la identificación o el nombre del contenedor para el contenedor que esté asociado con la imagen de hello-world y eliminarlo.

docker rm festive_williams

Puede iniciar un contenedor nuevo y nombrarlo usando el switch de --name. Además, puede usar el switch de --rm para crear un contenedor que se elimine automáticamente una vez que se detenga. Si desea aprender más sobre estas y otras opciones, consulte el comando docker run help.

Los contenedores se pueden convertir en imágenes que puede usar para crear contenedores nuevos. Vamos a ver cómo se hace eso.

Paso 7 — Hacer cambios en un contenedor a una imagen de Docker

Al iniciar una imagen de Docker, puede crear, modificar y borrar archivos al igual que lo hace con una máquina virtual. Los cambios que haga solamente se aplicarán a ese contenedor. Puede iniciarlo y detenerlo, pero una vez que lo destruya usando el comando docker rm, se perderán los cambios para siempre.

En esta sección, se le indica cómo guardar el estado de un contenedor como una imagen de Docker nueva.

Tras instalar Node.js dentro del contenedor de Ubuntu, tendrá un contenedor que se ejecuta de una imagen, pero el contenedor es distinto a la imagen que usó para crearlo. Tal vez quiera volver a usar este contenedor Node.js como base para imágenes nuevas más tarde.

Entonces, confirme los cambios en una instancia de imagen de Docker nueva usando el siguiente comando.

docker commit -m "What you did to the image" -a "Author Name" container_id repository/new_image_name

El switch **-m** es para el mensaje de confirmación que le ayuda a usted y a los demás a saber qué cambios hizo, mientras que **-a** se usa para especificar el autor. La container_id (identificación del contenedor) es la que anotó más temprano en el tutorial cuando inició la sesión interactiva de Docker. El repository suele ser su nombre de usuario de Docker Hub, a menos que haya creado repositorios adicionales en Docker Hub.

Por ejemplo, para el usuario **sammy**, cuya identificación de contenedor es d9b100f2f636, el comando sería:

docker commit -m "added Node.js" -a "sammy" d9b100f2f636 sammy/ubuntu-nodejs

Al *confirmar* una imagen, se guarda la imagen nueva localmente en su computadora. Más adelante en este tutorial, aprenderá cómo hacer push de una imagen a un registro de Docker como Docker Hub para que otros usuarios puedan tener acceso a la misma.

Si se listan las imágenes de Docker nuevamente, se mostrará la nueva imagen, al igual que la antigua de la que se derivó:

docker images

Verá un resultado como el siguiente:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sammy/ubuntu-nodejs	latest	7c1f35226ca6	7 seconds ago	179MB
ubuntu	latest	113a43faa138	4 weeks ago	81.2MB
hello-world	latest	e38bc07ac18e	2 months ago	1.85kB

En este ejemplo, la imagen nueva es ubuntu-nodejs, que se derivó de la imagen ubuntu existente de Docker Hub. La diferencia de tamaño refleja los cambios que se hicieron. Y en este ejemplo, el cambio fue que se instaló NodeJS. Por lo que, la próxima vez que deba ejecutar un contenedor usando Ubuntu con NodeJS preinstalado, simplemente puede usar la imagen nueva.

Además, puede crear Imágenes desde un Dockerfile, el cual le permite automatizar la instalación del software en una imagen nueva. No obstante, no se abarca eso en este tutorial.

Ahora, vamos a compartir la imagen nueva con los demás para que puedan crear contenedores usándola.

Paso 8 — Hacer push de imágenes de Docker a un repositorio Docker

El siguiente paso lógico tras crear una imagen nueva usando una imagen existente es compartirla con algunos amigos selectos, todo el mundo en Docker Hub u otro registro de Docker al que tenga acceso. Si desea hacer push de una imagen a Docker Hub o cualquier otro registro de Docker, debe tener una cuenta en ese sitio.

Esta sección le enseña a hacer push de una imagen Docker a Docker Hub. Consulte Cómo configurar un registro privado de Docker en Ubuntu 14.04 si desea aprender a crear su propio registro privado de Docker.

Primero, inicie sesión en Docker Hub para hacerle push a su imagen.

docker login -u docker-registry-username

Se le pedirá que se certifique utilizando su contraseña de Docker Hub. Si ingresó la contraseña correcta, la certificación debería se exitosa.

Nota: Si su nombre de usuario de registro de Docker es distinto al nombre de usuario local que usó para crear la imagen, deberá etiquetar su imagen con su nombre de usuario de registro. Para el ejemplo que se dio en el último paso, debe escribir:

docker tag sammy/ubuntu-nodejs docker-registry-username/ubuntu-nodejs

A continuación, podrá hacer el push de su propia imagen usando:

docker push docker-registry-username/docker-image-name

Para hacer el push de la imagen ubuntu-nodejs al repositorio de **sammy**, el comando sería:

docker push sammy/ubuntu-nodejs

Es posible que el proceso tarde un poco para terminarse a medida que se cargan las imágenes, pero una vez que se haya terminado, el resultado se verá así:

The push refers to a repository [docker.io/sammy/ubuntu-nodejs]

e3fbbfb44187: Pushed

5f70bf18a086: Pushed

a3b5c80a4eba: Pushed

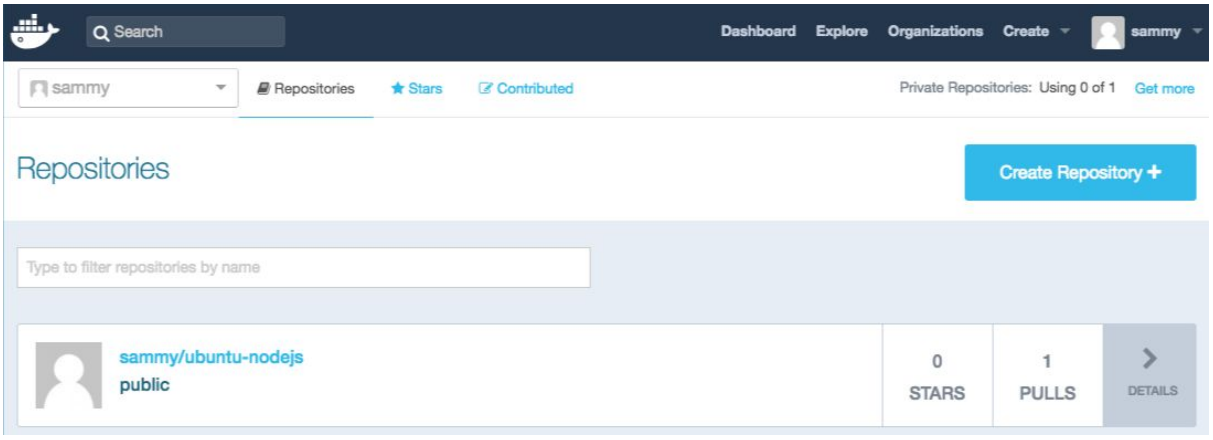
7f18b442972b: Pushed

3ce512daaf78: Pushed

7aae4540b42d: Pushed

...

Tras hacer push de una imagen al registro, debería aparecer en el panel de su cuenta, como se muestra en la imagen de abajo.



Si un intento de push le da un error de este tipo, seguramente no haya iniciado sesión:

The push refers to a repository [docker.io/sammy/ubuntu-nodejs]

e3fbbfb44187: Preparing

5f70bf18a086: Preparing

a3b5c80a4eba: Preparing

7f18b442972b: Preparing

3ce512daaf78: Preparing

7aae4540b42d: Waiting

unauthorized: authentication required

Inicie sesión usando el docker login y vuelva a intentar el push. Entonces, verifique que exista en su página del repositorio de Docker Hub.

Ahora puede usar docker pull sammy/ubuntu-nodejs para hacer el pull de la imagen a una nueva máquina y usarla para ejecutar un contenedor nuevo.

Otros datos de interés

Vídeo explicativo de Docker en inglés por Brad Traversy
https://www.youtube.com/watch?v=Kyx2PsuwomE

XAMPP en docker

Contenedor que dispone de xampp subido por tomsik68
https://hub.docker.com/r/tomsik68/xampp