COMANDOS DE UTILIZACION BASICOS DE DOCKER (LINUX Y WINDOWS)

USAR EL COMANDO DOCKER

El uso de `docker` consiste en pasar a este una cadena de opciones y comandos seguida de argumentos. La sintaxis adopta esta forma:

```
docker [option] [command] [arguments]
```

Copy

Para ver todos los subcomandos disponibles, escriba lo siguiente:

```
docker
```

Copy

A partir de Docker 19, en la lista completa de subcomandos disponibles se incluye lo siguiente:

```
Output
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
```

```
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by
default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
```

```
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit
codes
```

Si desea ver las opciones disponibles para un comando específico, escriba lo siguiente:

- `docker docker-subcommand --help`

Para ver información sobre Docker relacionada con todo el sistema, utilice lo siguiente:

- `docker info`

Exploremos algunos de estos comandos. Comenzaremos trabajando con imágenes.

Paso 4: Trabajar con imágenes de Docker

Los contenedores de Docker se construyen con imágenes de Docker. Por defecto, Docker obtiene estas imágenes de Docker Hub, un registro de Docker gestionado por Docker, la empresa responsable del proyecto Docker

. Cualquiera puede alojar sus imágenes en Docker Hub, de modo que la mayoría de las aplicaciones y las distribuciones de Linux que necesitará tendrán imágenes alojadas allí.

Para verificar si puede acceder a imágenes y descargarlas de Docker Hub, escriba lo siguiente:

- `docker run hello-world`

El resultado indicará que Docker funciona de forma correcta:

```
Output
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

...
```

Inicialmente, Docker no pudo encontrar la imagen de `hello-world` a nivel local. Por ello la descargó de Docker Hub, el repositorio predeterminado. Una vez que se descargó la imagen, Docker creó un contenedor a partir de ella y de la aplicación dentro del contenedor ejecutado, y mostró el mensaje.

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

```
docker search ubuntu
```

La secuencia de comandos rastreará Docker Hub y mostrará una lista de todas las imágenes cuyo nombre coincida con la cadena de búsqueda. En este caso, el resultado será similar a lo siguiente:

```
Output
```

```
NAME                                                    DESCRIPTION
STARS               OFFICIAL            AUTOMATED

ubuntu                                                  Ubuntu is a Debian-
based Linux operating sys…   10908               [OK]

dorowu/ubuntu-desktop-lxde-vnc                          Docker image to provide
HTML5 VNC interface …   428                                 [OK]

rastasheep/ubuntu-sshd                                  Dockerized SSH service,
built on top of offi…   244                                 [OK]

consol/ubuntu-xfce-vnc                                  Ubuntu container with
"headless" VNC session…   218                                 [OK]

ubuntu-upstart                                          Upstart is an event-
based replacement for th…   108                 [OK]

ansible/ubuntu14.04-ansible                             Ubuntu 14.04 LTS with

...
```

En la columna de **OFFICIAL**, **OK** indica una imagen creada y avalada por la empresa responsable del proyecto. Una vez que identifique la imagen que desearía usar, puede descargarla a su computadora usando el subcomando `pull`.

Ejecute el siguiente comando para descargar la imagen oficial de `ubuntu` a su ordenador:

```
docker pull ubuntu
```

Verá el siguiente resultado:

```
Output

Using default tag: latest

latest: Pulling from library/ubuntu

d51af753c3d3: Pull complete

fc878cd0a91c: Pull complete

6154df8ff988: Pull complete
```

```
fee5db0ff82f: Pull complete
Digest: sha256:747d2dbbaaee995098c9792d99bd333c6783ce56150d1b11e333bbceed5c54d7
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

Una vez descargada una imagen, puede ejecutar un contenedor usando la imagen descargada con el subcomando `run`. Como pudo ver en el ejemplo de `hello-world`, si no se descargó una imagen al ejecutarse `docker` con el subcomando `run`, el cliente de Docker descargará primero la imagen y luego ejecutará un contenedor utilizándola.

Para ver las imágenes descargadas a su computadora, escriba lo siguiente:

```
docker images
```

El resultado debe tener un aspecto similar al siguiente:

```
Output
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              latest              1d622ef86b13        3 weeks ago
73.9MB
hello-world         latest              bf756fb1ae65        4 months ago


13.3kB
```

Como verá más adelante en este tutorial, las imágenes que utilice para ejecutar contenedores pueden modificarse y utilizarse para generar nuevas imágenes que se pueden cargar posteriormente (*"introducir"* es el término técnico) a Docker Hub o a otros registros de Docker.

Veamos en mayor profundidad la forma de ejecutar los contenedores.

PASO 5: EJECUTAR UN CONTENEDOR DE DOCKER

El contenedor `hello-world` que ejecutó en el paso anterior es un ejemplo de un contenedor que se ejecuta y se cierra tras emitir un mensaje de prueba. Los contenedores pueden ofrecer una utilidad mucho mayor y ser interactivos. Después de todo, son similares a las máquinas virtuales, aunque más flexibles con los recursos.

Como ejemplo, ejecutemos un contenedor usando la imagen más reciente de Ubuntu. La combinación de los conmutadores **-i** y **-t** le proporcionan un acceso interactivo del shell al contenedor:

```
docker run -it ubuntu
```

Su símbolo del sistema debe cambiar para reflejar el hecho de que ahora trabaja dentro del contenedor y debe adoptar esta forma:

```
Output

root@d9b100f2f636:/#
```

Tenga en cuenta el ID del contenedor en el símbolo del sistema. En este ejemplo, es `d9b100f2f636`. Más adelante, necesitará ese ID de contenedor para identificar el contenedor cuando desee eliminarlo.

Ahora puede ejecutar cualquier comando dentro del contenedor. Por ejemplo, actualicemos la base de datos del paquete dentro del contenedor. No es necesario prefijar ningún comando con `sudo`, ya que realiza operaciones dentro del contenedor como el usuario **root**:

```
apt update
```

Luego, instale cualquier aplicación en él. Probemos con Node.js:

```
apt install nodejs
```

Con esto, se instala Node.js en el contenedor desde el repositorio oficial de Ubuntu. Cuando finalice la instalación, verifique que Node.js esté instalado:

```
node -v
```

Verá el número de versión en su terminal:

```
Output
v10.19.0
```

Cualquier cambio que realice dentro del contenedor solo se aplica a este.

Para cerrar el contenedor, escriba `exit` a línea de comandos.

A continuación, veremos la forma de administrar los contenedores en nuestro sistema.

PASO 6: ADMINISTRAR CONTENEDORES DE DOCKER

Después de usar Docker durante un tiempo, tendrá muchos contenedores activos (en ejecución) e inactivos en su computadora. Para ver **los activos**, utilice lo siguiente:

```
docker ps
```

Verá una salida similar a la siguiente:

```
Output
CONTAINER ID        IMAGE               COMMAND             CREATED
```

A través de este tutorial, inició dos contenedores: uno de la imagen `hello-world` y otro de la imagen `ubuntu`. Ambos contenedores ya no están en ejecución, pero aún existen en su sistema.

Para ver todos los contenedores, activos e inactivos, ejecute `docker ps` con el conmutador `-a`:

```
docker ps -a
```

Visualizará un resultado similar a esto:

```
1c08a7a0d0e4        ubuntu              "/bin/bash"         2 minutes ago
Exited (0) 8 seconds ago                       quizzical_mcnulty
a707221a5f6c        hello-world         "/hello"            6 minutes ago
Exited (0) 6 minutes ago                       youthful_curie
```

Para ver el último contenedor que creó, páselo al conmutador `-l`:

```
docker ps -l
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES

1c08a7a0d0e4        ubuntu              "/bin/bash"         2 minutes ago
Exited (0) 40 seconds ago                       quizzical_mcnulty
```

Para iniciar un contenedor detenido, utilice `docker start`, seguido del o el nombre ID del contenedor. Vamos a iniciar el contenedor basado en Ubuntu con el ID `1c08a7a0d0e4`:

```
docker start 1c08a7a0d0e4
```

El contenedor se iniciará y podrá usar `docker ps` para ver su estado:

```
Output
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
1c08a7a0d0e4        ubuntu              "/bin/bash"         3 minutes ago
Up 5 seconds                            quizzical_mcnulty
```

Para detener un contenedor en funcionamiento, utilice `docker stop`, seguido del ID o nombre del contenedor. Esta vez usaremos el nombre que Docker asignó al contenedor, que es `quizzical_mcnulty`:

```
docker stop quizzical_mcnulty
```

Una vez que decidida que ya no necesita un contenedor, elimínelo con el comando `docker rm` y use nuevamente el ID o el nombre del contenedor. Utilice el comando `docker ps -a` para encontrar el ID o nombre del contenedor asociado con la imagen `hello-world` y elimínelo.

```
docker rm youthful_curie
```

Puede iniciar un nuevo contenedor y darle un nombre usando el conmutador `--name`. También podrá usar el conmutador de `--rm` para crear un contenedor que se elimine de forma automática cuando se detenga. Consulte el comando `docker run help` para obtener más información sobre estas y otras opciones.

Los contenedores pueden convertirse en imágenes que podrá usar para crear contenedores nuevos. Veamos cómo funciona esto.

PASO 7: CONFIRMAR CAMBIOS APLICADOS A UNA IMAGEN DE DOCKER EN UN CONTENEDOR

Cuando inicie una imagen de Docker, podrá crear, modificar y eliminar archivos del mismo modo que con una máquina virtual. Los cambios que realice solo se aplicarán al contenedor en cuestión. Podrá iniciarlo y detenerlo, pero una vez que lo destruya con el comando `docker rm`, los cambios se perderán por completo.

En esta sección verá la forma de guardar el estado de un contenedor como una nueva imagen de Docker.

Después de instalar Node.js dentro del contenedor de Ubuntu, dispondrá de un contenedor que se ejecuta a partir de una imagen, pero este es diferente de la imagen que utilizó para crearlo. Sin embargo, quizá desee reutilizar este contenedor Node.js como base de nuevas imágenes posteriormente.

Luego, confirme los cambios en una nueva instancia de imagen de Docker utilizando el siguiente comando:

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

El conmutador **-m** es para el mensaje de confirmación que le permite a usted y a otros a saber qué cambios realizaron, mientras que **-a** se utiliza para especificar el autor. El `container_id` es el que observó anteriormente en el tutorial cuando inició la sesión interactiva de Docker. A menos que haya creado repositorios adicionales en Docker Hub, `repository` generalmente corresponde a su nombre de usuario de Docker Hub.

Por ejemplo, para el usuario **sammy**, con el ID de contenedor `d9b100f26f636`, el comando sería el siguiente:

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

Cuando confirme una imagen, la nueva imagen se guardará a nivel local en su computadora. Más adelante, en este tutorial, aprenderá a introducir una imagen en un registro de Docker como Docker Hub para que otros puedan acceder a ella.

Listar las imágenes de Docker de nuevo mostrará la nueva imagen, así como la anterior de la que se derivó:

```
docker images
```

Verá resultados como este:

```
Output
REPOSITORY              TAG                 IMAGE ID            CREATED
SIZE
sammy/ubuntu-nodejs   latest              7c1f35226ca6        7 seconds ago
179MB
...
```

En este ejemplo `ubuntu-nodejs` es la nueva imagen, derivada de la imagen `ubuntu` existente de Docker Hub. La diferencia de tamaño refleja los cambios realizados.

En este ejemplo, el cambio fue la instalación de NodeJS. Por ello, la próxima vez que deba ejecutar un contenedor usando Ubuntu con NodeJS preinstalado, podrá usar simplemente la nueva imagen.

También podrá crear imágenes de un `Dockerfile`, lo cual le permitirá automatizar la instalación de software en una nueva imagen. Sin embargo, eso queda fuera del alcance de este tutorial.

AHORA, COMPARTIREMOS LA NUEVA IMAGEN CON TERCEROS PARA QUE PUEDAN CREAR CONTENEDORES A PARTIR DE ELLA.

PASO 8: INTRODUCIR IMÁGENES DE DOCKER EN UN REPOSITORIO DE DOCKER

El siguiente paso lógico después de crear una nueva imagen a partir de una imagen existente es compartirla con algunos de sus amigos, con todo el mundo en Docker Hub, o en otro registro de Docker al que tenga acceso. Para introducir una imagen a Docker Hub o a cualquier otro registro de Docker, deberá tener una cuenta en el sistema.

Para introducir su imagen, primero inicie sesión en Docker Hub.

```
docker login -u docker-registry-username
```

Se le solicitará autenticarse usando su contraseña de Docker Hub. Si especificó la contraseña correcta, la autenticación tendrá éxito.

Nota: Si su nombre de usuario de registro de Docker es diferente del 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 muestra en el último paso, deberá escribir lo siguiente:

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

Luego podrá introducir su propia imagen usando lo siguiente:

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

Para introducir la imagen `ubuntu-nodejs` en el repositorio de **sammy**, el comando sería el siguiente:

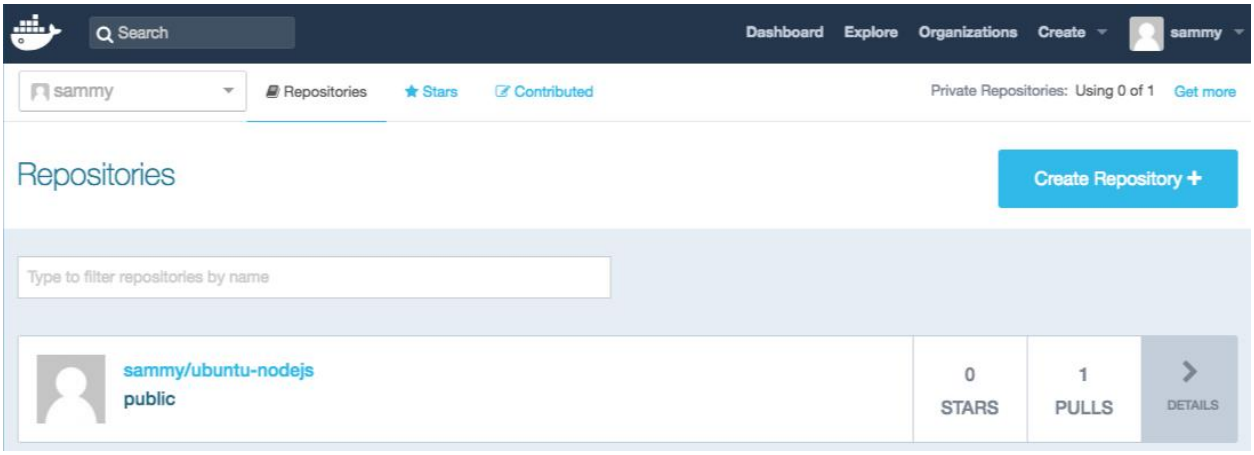
```
docker push sammy/ubuntu-nodejs
```

El proceso puede tardar un tiempo en completarse cuando se suben las imágenes, pero una vez que finalice el resultado será el siguiente:

```
Output
The push refers to a repository [docker.io/sammy/ubuntu-nodejs]
e3fbbfb44187: Pushed
5f70bf18a086: Pushed
a3b5c80a4eba: Pushed
7f18b442972b: Pushed
3ce512daaf78: Pushed
7aae4540b42d: Pushed


...
```

Una vez que introduce una imagen en un registro, esta debe aparecer en el panel de su cuenta, como se muestra en la siguiente captura:



Si un intento de introducción produce un error de este tipo, es probable que no haya iniciado sesión:

```
Output
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 con docker login` y repita el intento de introducción. Luego, compruebe que exista en su página de repositorios de Docker Hub.

Ahora podrá usar `docker pull sammy/ubuntu-nodejs` para introducir la imagen en una nueva máquina y usarla para ejecutar un nuevo contenedor.