

Guía de laboratorio 1 – Comandos básicos de Docker

Introducción

En esta guía de laboratorio se abordarán los comandos básicos de Docker para realizar acciones como verificar versión de Docker instalada en el equipo, listar los contenedores en ejecución, conocer la cantidad de imágenes disponibles, ejecutar, detener y eliminar contenedores, descargar y eliminar imágenes, etc., tiene como objetivo ayudar a los estudiantes a comprender y familiarizarse con los comandos que permiten realizar las acciones básicas de Docker.

Duración estimada en la realización de la guía

45 minutos.

Desarrollo de la guía

1) ¿Cuál es la versión del servidor Docker que se ejecuta en el host?

Ejecute el comando **docker version** para saber la versión del servidor de Docker.

```
Terminal 1 +
$ docker version
Client: Docker Engine - Community
Version:      19.03.15
API version:  1.40
Go version:   go1.13.15
Git commit:   99e3ed8919
Built:        Sat Jan 30 03:17:11 2021
OS/Arch:      linux/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version:      19.03.15
API version:  1.40 (minimum version 1.12)
Go version:   go1.13.15
Git commit:   99e3ed8919
Built:        Sat Jan 30 03:15:40 2021
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.4.3
GitCommit:    269548fa27e0089a8b8278fc4fc781d7f65a939b
runc:
Version:      1.0.0-rc92
GitCommit:    ff819c7e9184c13b7c2607fe6c30ae19403a7aff
docker-init:
Version:      0.18.0
GitCommit:    fec3683
$
```

Ilustración 1: Verificación de la versión de Docker

La versión del servidor Docker es la 19.03.15.

2) ¿Cuántos contenedores se ejecutan en este host?

Ejecute el comando **docker ps** para ver la cantidad de contenedores en ejecución.

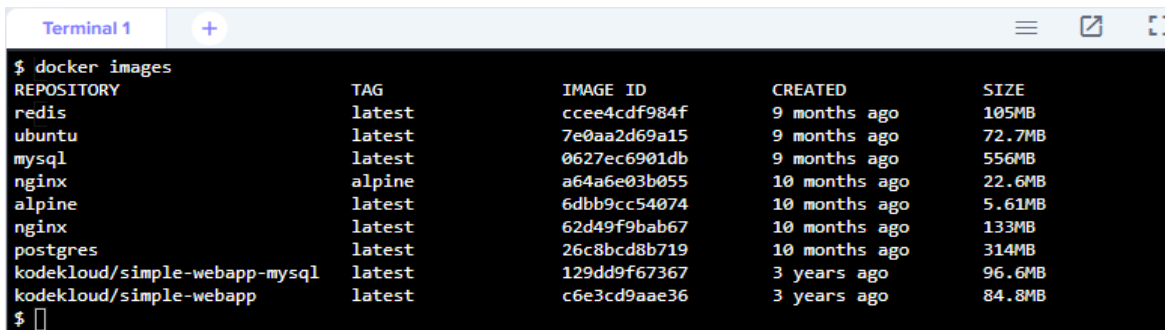
```
Terminal 1 +
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
$
```

Ilustración 2: Verificación de la cantidad de contenedores en ejecución

Como se observa no hay contenedores en ejecución.

3) ¿Cuántas imágenes hay disponibles en este host?

Ejecute el comando **docker images** para ver la cantidad de imágenes disponibles.



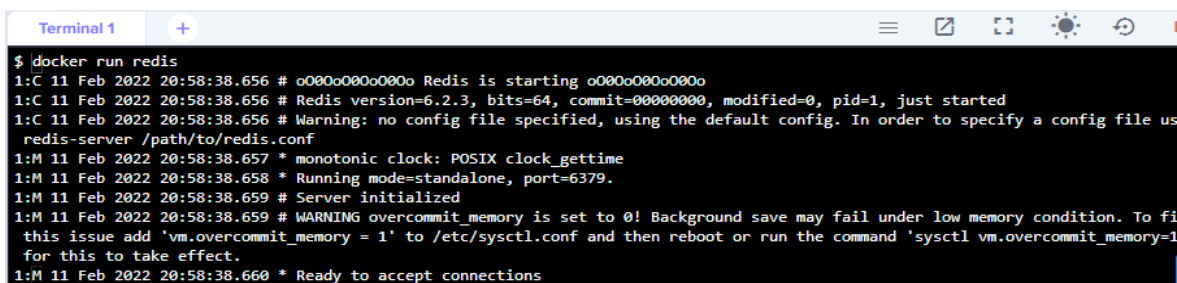
```
Terminal 1
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis                latest              ccee4cdf984f       9 months ago       105MB
ubuntu              latest              7e0aa2d69a15       9 months ago       72.7MB
mysql               latest              0627ec6901db       9 months ago       556MB
nginx               alpine             a64a6e03b055       10 months ago      22.6MB
alpine              latest              6dbb9cc54074       10 months ago      5.61MB
nginx               latest              62d49f9bab67       10 months ago      133MB
postgres            latest              26c8bcd8b719       10 months ago      314MB
kodekloud/simple-webapp-mysql latest              129dd9f67367       3 years ago         96.6MB
kodekloud/simple-webapp latest              c6e3cd9aae36       3 years ago         84.8MB
$
```

Ilustración 3: Verificación de la cantidad de imágenes disponibles

En este host hay 9 imágenes disponibles.

4) Ejecute un contenedor usando la imagen redis

Ejecute el comando **docker run redis**



```
Terminal 1
$ docker run redis
1:C 11 Feb 2022 20:58:38.656 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 11 Feb 2022 20:58:38.656 # Redis version=6.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 11 Feb 2022 20:58:38.656 # Warning: no config file specified, using the default config. In order to specify a config file use
redis-server /path/to/redis.conf
1:M 11 Feb 2022 20:58:38.657 * monotonic clock: POSIX clock_gettime
1:M 11 Feb 2022 20:58:38.658 * Running mode=standalone, port=6379.
1:M 11 Feb 2022 20:58:38.659 # Server initialized
1:M 11 Feb 2022 20:58:38.659 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix
this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1'
for this to take effect.
1:M 11 Feb 2022 20:58:38.660 * Ready to accept connections
```

Ilustración 4: Ejecución de un contenedor usando la imagen redis

5) Detenga el contenedor que acaba de crear.

Para detener un contenedor en ejecución utilice la combinación de teclas CTL+C

```
Terminal 1 +
$ docker run redis
1:C 11 Feb 2022 20:58:38.656 # o000o000o000o Redis is starting o000o000o000o
1:C 11 Feb 2022 20:58:38.656 # Redis version=6.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 11 Feb 2022 20:58:38.656 # Warning: no config file specified, using the default config. In order to specify a config file use
redis-server /path/to/redis.conf
1:M 11 Feb 2022 20:58:38.657 * monotonic clock: POSIX clock_gettime
1:M 11 Feb 2022 20:58:38.658 * Running mode=standalone, port=6379.
1:M 11 Feb 2022 20:58:38.659 # Server initialized
1:M 11 Feb 2022 20:58:38.659 # WARNING overcommit memory is set to 0! Background save may fail under low memory condition. To fix
this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1'
for this to take effect.
1:M 11 Feb 2022 20:58:38.660 * Ready to accept connections
^C1:signal-handler (1644613207) Received SIGINT scheduling shutdown...
1:M 11 Feb 2022 21:00:07.383 # User requested shutdown...
1:M 11 Feb 2022 21:00:07.383 * Saving the final RDB snapshot before exiting.
1:M 11 Feb 2022 21:00:07.386 * DB saved on disk
1:M 11 Feb 2022 21:00:07.386 # Redis is now ready to exit, bye bye...
$
```

Ilustración 5: Deteniendo contenedor

6) ¿Cuántos contenedores se están ejecutando en este host ahora?

Ejecute el comando **docker ps** para ver la cantidad de contenedores en ejecución.

```
Terminal 1 +
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
e9d0ab52e48c   alpine        "sleep 1000"            54 seconds ago Up 52 seconds        sharp_poincare
12f36004321e   nginx:alpine  "/docker-entrypoint..." 56 seconds ago Up 54 seconds        80/tcp        nginx-2
8e3105aad953   nginx:alpine  "/docker-entrypoint..." 58 seconds ago Up 56 seconds        80/tcp        nginx-1
c3806857e677   ubuntu        "sleep 1000"            About a minute ago Up 58 seconds        awesome_northcut
$
```

Ilustración 6: Cantidad de contenedores en ejecución

Hay 4 contenedores en ejecución

7) ¿Cuántos contenedores están PRESENTES en el host ahora? (Incluidos tanto los que se ejecutan como los que no se ejecutan)

Para ver la cantidad de contenedores existentes (ejecutándose o no) utilice el comando **docker ps -a**

Terminal 1						
+						
\$ docker ps -a						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9562ea0f7dba	alpine	"/bin/sh"	3 minutes ago	Exited (0) 3 minutes ago		reverent_liskov
e9d0ab52e48c	alpine	"sleep 1000"	3 minutes ago	Up 3 minutes		sharp_poincare
12f36004321e	nginx:alpine	"/docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	80/tcp	nginx-2
8e3105aad953	nginx:alpine	"/docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	80/tcp	nginx-1
c3806857e677	ubuntu	"sleep 1000"	3 minutes ago	Up 3 minutes		awesome_northcut
ff2c126ebb68	redis	"docker-entrypoint.s..."	4 minutes ago	Exited (0) 4 minutes ago		interesting_goldst
ine						
ee565dcfb489	redis	"docker-entrypoint.s..."	4 minutes ago	Exited (0) 4 minutes ago		crazy_curran
\$						

Ilustración 7: Verificación de contenedores existentes

En el host hay 7 contenedores.

8) ¿Cuál es la imagen utilizada para ejecutar el contenedor nginx-1?

Ejecute el comando **docker ps** y verifique debajo de la columna "IMAGE".

Terminal 1						
+						
\$ docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e9d0ab52e48c	alpine	"sleep 1000"	4 minutes ago	Up 4 minutes		sharp_poincare
12f36004321e	nginx:alpine	"/docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	80/tcp	nginx-2
8e3105aad953	nginx:alpine	"/docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	80/tcp	nginx-1
c3806857e677	ubuntu	"sleep 1000"	5 minutes ago	Up 4 minutes		awesome_northcut
\$						

Ilustración 8: Verificación de la imagen que utiliza un contenedor

La imagen utilizada en el contenedor nginx-1 es nginx:alpine.

9) ¿Cuál es el nombre del contenedor creado usando la imagen de ubuntu?

Ejecute el comando **docker ps** y observe la columna "NAMES".

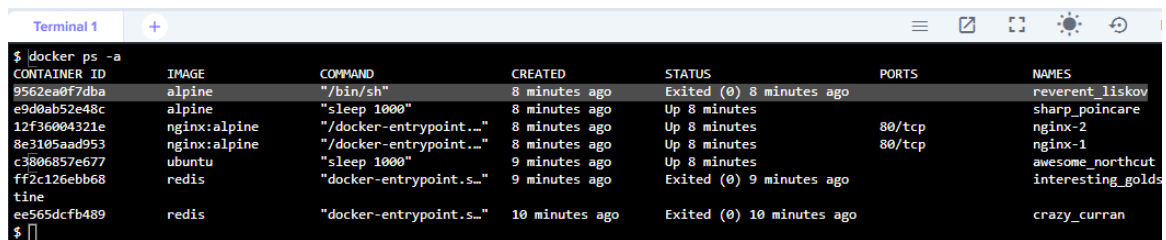
Terminal 1						
+						
\$ docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e9d0ab52e48c	alpine	"sleep 1000"	7 minutes ago	Up 7 minutes		sharp_poincare
12f36004321e	nginx:alpine	"/docker-entrypoint.s..."	7 minutes ago	Up 7 minutes	80/tcp	nginx-2
8e3105aad953	nginx:alpine	"/docker-entrypoint.s..."	7 minutes ago	Up 7 minutes	80/tcp	nginx-1
c3806857e677	ubuntu	"sleep 1000"	7 minutes ago	Up 7 minutes		awesome_northcut
\$						

Ilustración 9: Verificación del nombre de un contenedor

El nombre del contenedor creado es Awesome_northcut.

10) ¿Cuál es el ID del contenedor que usa la imagen alpine y no se está ejecutando?

Ejecute el comando **docker ps -a** e identifique el ID del contenedor que usa la imagen alpine.



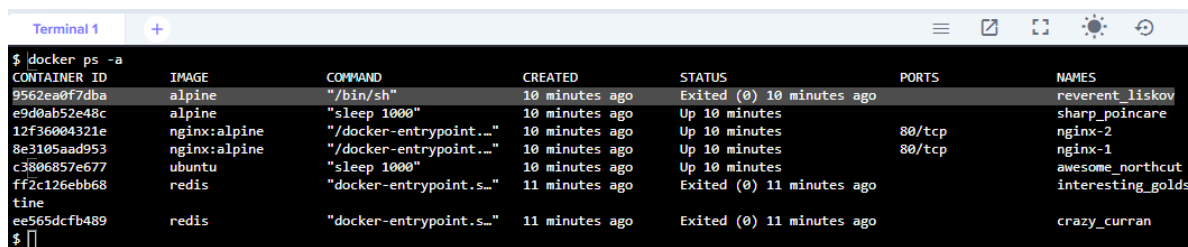
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9562ea0f7dba	alpine	"/bin/sh"	8 minutes ago	Exited (0) 8 minutes ago		reverent_liskov
e9d0ab52e48c	alpine	"sleep 1000"	8 minutes ago	Up 8 minutes		sharp_poincare
12f36004321e	nginx:alpine	"/docker-entrypoint..."	8 minutes ago	Up 8 minutes	80/tcp	nginx-2
8e3105aad953	nginx:alpine	"/docker-entrypoint..."	8 minutes ago	Up 8 minutes	80/tcp	nginx-1
c3806857e677	ubuntu	"sleep 1000"	9 minutes ago	Up 8 minutes		awesome_northcut
ff2c126ebb68	redis	"docker-entrypoint.s..."	9 minutes ago	Exited (0) 9 minutes ago		interesting_gold
tine						
ee565dcfb489	redis	"docker-entrypoint.s..."	10 minutes ago	Exited (0) 10 minutes ago		crazy_curran

Ilustración 10: Verificación del ID de un contenedor

Dicho contenedor tiene el ID 9562ea0f7dba.

11) ¿Cuál es el estado del contenedor alpine detenido?

Ejecute el comando **docker ps -a** y observe la columna "STATUS".



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9562ea0f7dba	alpine	"/bin/sh"	10 minutes ago	Exited (0) 10 minutes ago		reverent_liskov
e9d0ab52e48c	alpine	"sleep 1000"	10 minutes ago	Up 10 minutes		sharp_poincare
12f36004321e	nginx:alpine	"/docker-entrypoint..."	10 minutes ago	Up 10 minutes	80/tcp	nginx-2
8e3105aad953	nginx:alpine	"/docker-entrypoint..."	10 minutes ago	Up 10 minutes	80/tcp	nginx-1
c3806857e677	ubuntu	"sleep 1000"	10 minutes ago	Up 10 minutes		awesome_northcut
ff2c126ebb68	redis	"docker-entrypoint.s..."	11 minutes ago	Exited (0) 11 minutes ago		interesting_gold
tine						
ee565dcfb489	redis	"docker-entrypoint.s..."	11 minutes ago	Exited (0) 11 minutes ago		crazy_curran

Ilustración 11: Verificación del estado de un contenedor

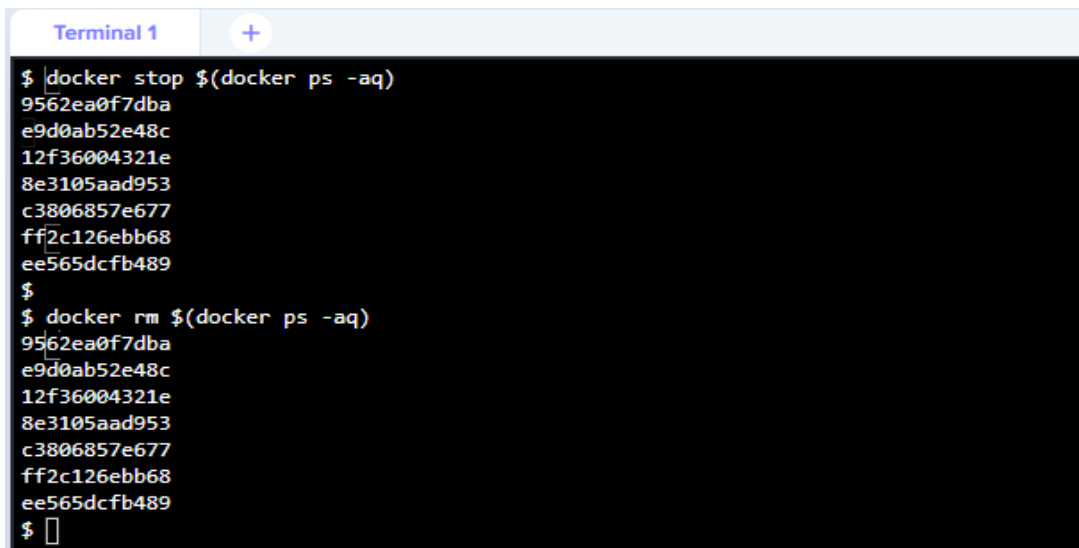
El estado de dicho contenedor es Exited, lo cual indica que no está en ejecución.

12) Elimine todos los contenedores del Docker Host, tanto los que se ejecutan como los que no se ejecutan.

Nota: recuerde que es posible que deba detener los contenedores antes de eliminarlos.

Ejecute el comando **docker stop \$(docker ps -aq)** para detener todos los contenedores a la vez.

Ejecute el comando **docker rm \$(docker ps -aq)** para eliminar todos los contenedores detenidos a la vez.

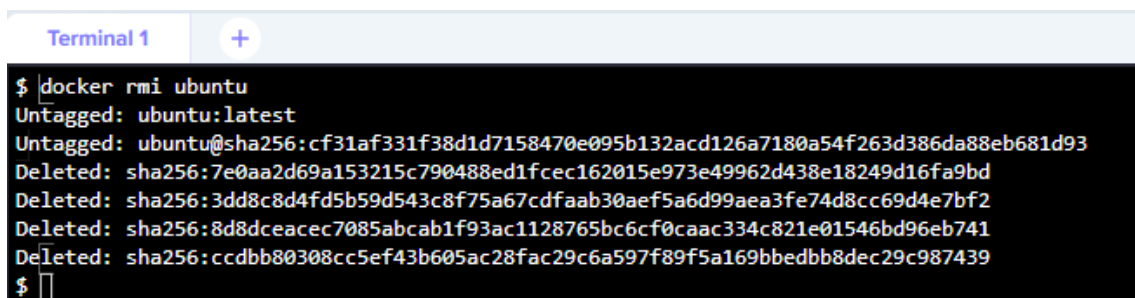


```
Terminal 1 +
$ docker stop $(docker ps -aq)
9562ea0f7dba
e9d0ab52e48c
12f36004321e
8e3105aad953
c3806857e677
ff2c126ebb68
ee565dcfb489
$
$ docker rm $(docker ps -aq)
9562ea0f7dba
e9d0ab52e48c
12f36004321e
8e3105aad953
c3806857e677
ff2c126ebb68
ee565dcfb489
$
```

Ilustración 12: Detención y eliminación de todos los contenedores

13) Elimine la imagen de Ubuntu

Ejecute el comando **docker rmi Ubuntu**

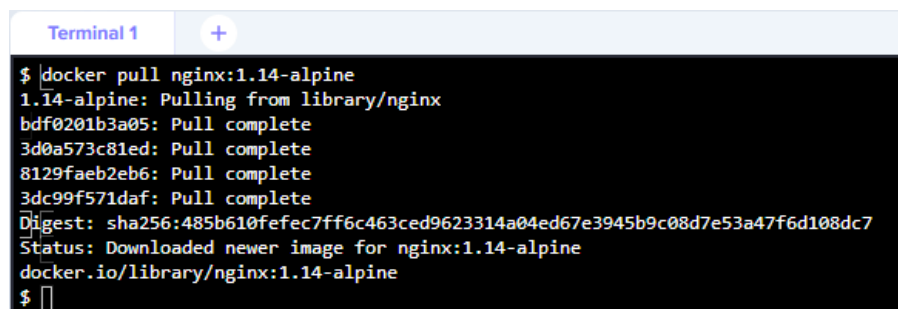


```
Terminal 1 +
$ docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:cf31af331f38d1d7158470e095b132acd126a7180a54f263d386da88eb681d93
Deleted: sha256:7e0aa2d69a153215c790488ed1fcec162015e973e49962d438e18249d16fa9bd
Deleted: sha256:3dd8c8d4fd5b59d543c8f75a67cdfaab30aef5a6d99aea3fe74d8cc69d4e7bf2
Deleted: sha256:8d8dceacec7085abcb1f93ac1128765bc6cf0caac334c821e01546bd96eb741
Deleted: sha256:ccd8b80308cc5ef43b605ac28fac29c6a597f89f5a169bbdbb8dec29c987439
$
```

Ilustración 13: Eliminación de una imagen Docker

14) Extraer una imagen que se utilizará para ejecutar un contenedor más adelante. Descargue la imagen `nginx:1.14-alpine`, sólo descargue dicha imagen, no cree un contenedor.

Ejecute el comando **`docker pull nginx:1.14-alpine`**

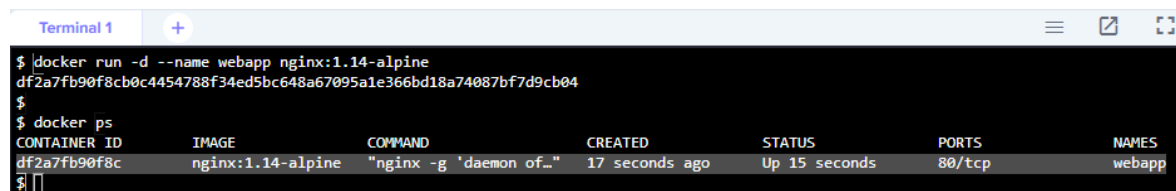


```
Terminal 1 +
$ docker pull nginx:1.14-alpine
1.14-alpine: Pulling from library/nginx
bdf0201b3a05: Pull complete
3d0a573c81ed: Pull complete
8129faeb2eb6: Pull complete
3dc99f571daf: Pull complete
Digest: sha256:485b610fefec7ff6c463ced9623314a04ed67e3945b9c08d7e53a47f6d108dc7
Status: Downloaded newer image for nginx:1.14-alpine
docker.io/library/nginx:1.14-alpine
$
```

Ilustración 14: Descargar una imagen Docker

15) Ejecute un contenedor con la imagen `nginx:1.14-alpine` y asígnele el nombre `webapp`

Ejecute el comando **`docker run -d --name webapp nginx:1.14-alpine`** y verifique el estado del contenedor creado por el comando **`docker ps`**.



```
Terminal 1 +
$ docker run -d --name webapp nginx:1.14-alpine
df2a7fb90f8cb0c4454788f34ed5bc648a67095a1e366bd18a74087bf7d9cb04
$
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
df2a7fb90f8c       nginx:1.14-alpine  "nginx -g 'daemon of..." 17 seconds ago      Up 15 seconds      80/tcp             webapp
$
```

Ilustración 15: Ejecución y asignación de nombre a un contenedor

El contenedor fue creado exitosamente y se encuentra en ejecución. El indicador `-d` permite ejecutar el contenedor en segundo plano.

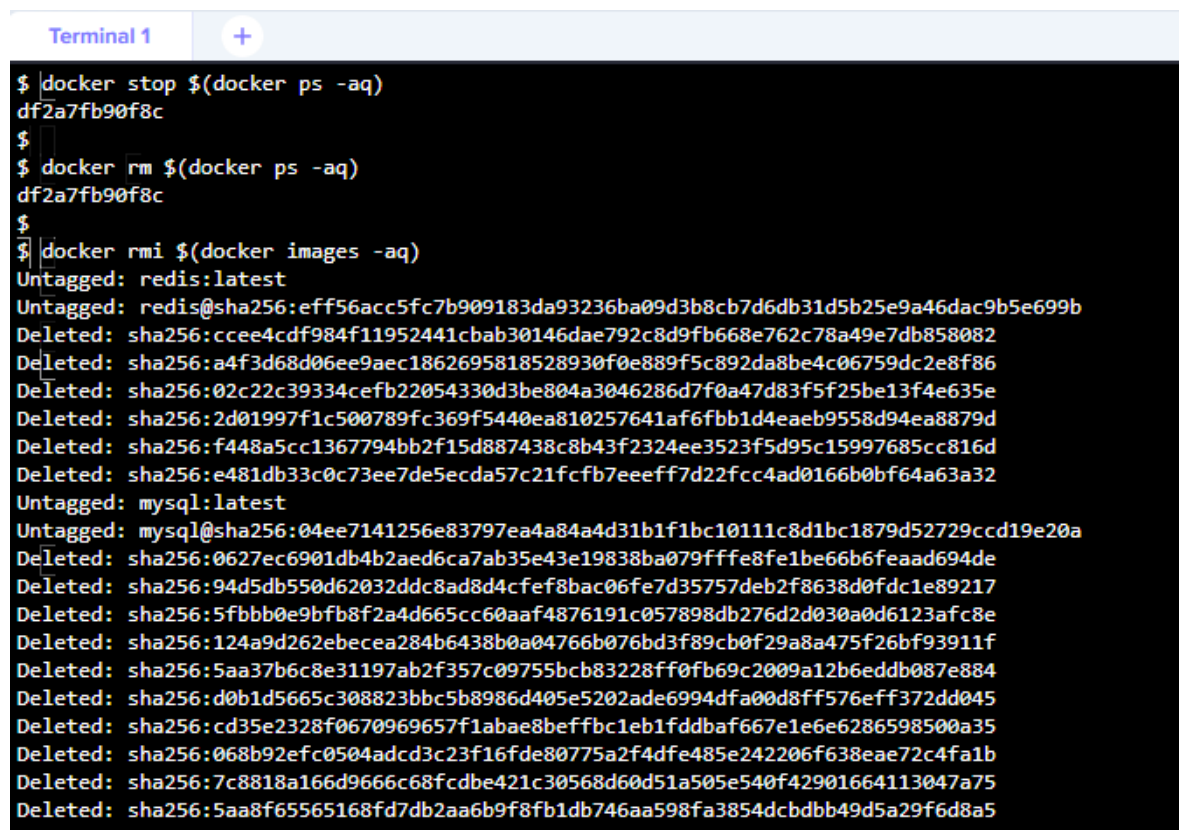
16) Elimine todas las imágenes en el host, retire los contenedores según sea necesario.

Nota: Es importante que detenga y elimine todos los contenedores que utilizan las imágenes.

Ejecute el comando **docker stop \$(docker ps -aq)** para detener todos los contenedores.

Ejecute el comando **docker rm \$(docker ps -aq)** para eliminar todos los contenedores.

Por último, ejecute el comando **docker rmi \$(docker images -aq)** para eliminar todas las imágenes disponibles.



```
Terminal 1
$ docker stop $(docker ps -aq)
df2a7fb90f8c
$
$ docker rm $(docker ps -aq)
df2a7fb90f8c
$
$ docker rmi $(docker images -aq)
Untagged: redis:latest
Untagged: redis@sha256:eff56acc5fc7b909183da93236ba09d3b8cb7d6db31d5b25e9a46dac9b5e699b
Deleted: sha256:ccee4cdf984f11952441cbab30146dae792c8d9fb668e762c78a49e7db858082
Deleted: sha256:a4f3d68d06ee9aec1862695818528930f0e889f5c892da8be4c06759dc2e8f86
Deleted: sha256:02c22c39334cefb22054330d3be804a3046286d7f0a47d83f5f25be13f4e635e
Deleted: sha256:2d01997f1c500789fc369f5440ea810257641af6fbb1d4eae9558d94ea8879d
Deleted: sha256:f448a5cc1367794bb2f15d887438c8b43f2324ee3523f5d95c15997685cc816d
Deleted: sha256:e481db33c0c73ee7de5ecd5a57c21fcfb7eeeff7d22fcc4ad0166b0bf64a63a32
Untagged: mysql:latest
Untagged: mysql@sha256:04ee7141256e83797ea4a84a4d31b1f1bc10111c8d1bc1879d52729ccd19e20a
Deleted: sha256:0627ec6901db4b2aed6ca7ab35e43e19838ba079fffe8fe1be66b6feaad694de
Deleted: sha256:94d5db550d62032ddc8ad8d4cfef8bac06fe7d35757deb2f8638d0fdc1e89217
Deleted: sha256:5fbbb0e9bfb8f2a4d665cc60aaf4876191c057898db276d2d030a0d6123afc8e
Deleted: sha256:124a9d262ebecea284b6438b0a04766b076bd3f89cb0f29a8a475f26bf93911f
Deleted: sha256:5aa37b6c8e31197ab2f357c09755bcb83228ff0fb69c2009a12b6eddb087e884
Deleted: sha256:d0b1d5665c308823bbc5b8986d405e5202ade6994dfa00d8ff576eff372dd045
Deleted: sha256:cd35e2328f0670969657f1abae8beffbc1eb1fddbafe667e1e6e6286598500a35
Deleted: sha256:068b92efc0504adcd3c23f16fde80775a2f4dfe485e242206f638eae72c4fa1b
Deleted: sha256:7c8818a166d9666c68fcdbe421c30568d60d51a505e540f42901664113047a75
Deleted: sha256:5aa8f65565168fd7db2aa6b9f8fb1db746aa598fa3854dcdbbb49d5a29f6d8a5
```

Ilustración 16: Eliminación de todas las imágenes de Docker.