EJERCICIOS DOCKER

1) Realizar DOCUMENTOS (ejercicios\_docker.md) con las instrucciones que se han seguido en cada uno de los siguientes tutoriales online. (elegir a o b)

Ejercicios Oficiales de DOCKER (<https://training.play-with-docker.com/> )

a) Para Administradores:

[Your First Linux Containers](https://training.play-with-docker.com/ops-s1-hello): In this lab you will explore the basics of running containers.

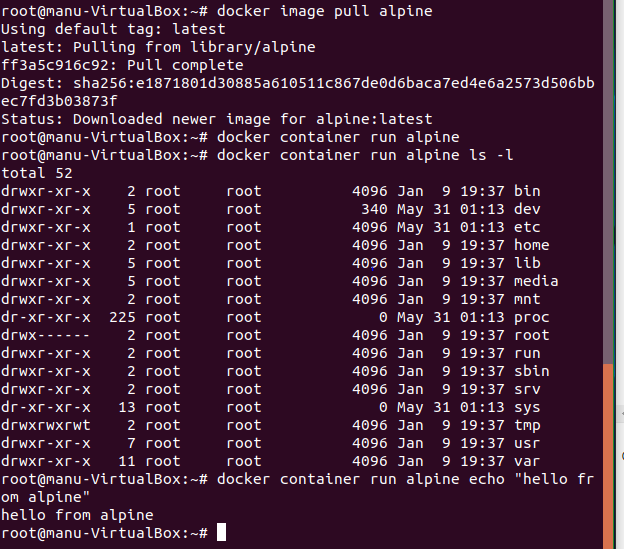
* Imagenes de Docker:

Alpine es una distribución liviana de Linux por lo que es rápido de desplegar y ejecutar, por lo que es punto de partida popular para muchas otras imágenes.

Para comenzar, ejecutemos lo siguiente en nuestra terminal:

**“docker image pull alpine”**

El **Pull** recupera la imagen alpina del registro Docker y la guarda en nuestro sistema.



El hecho de que el contenedor salga después de ejecutar nuestro comando importante, probemos algo más emocionante.

**docker container run alpine echo "hello from alpine"**

El siguiente comando:

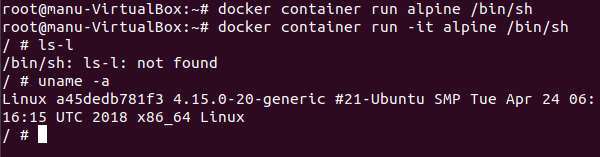
**“docker container run alpine /bin/sh”** no sale nada porque comenzó una tercera instancia del contenedor alpini y ejecuto el comando /bin/sh y luego salió.

Para que salga introducimos:

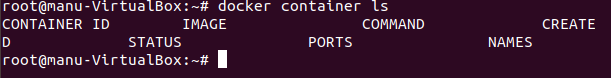
**“docker container run -it alpine /bin/sh”**

Ahora está dentro del contenedor ejecutando un Shell de Linux y puede probar algunos comandos como **uname -a**

Para salir exit

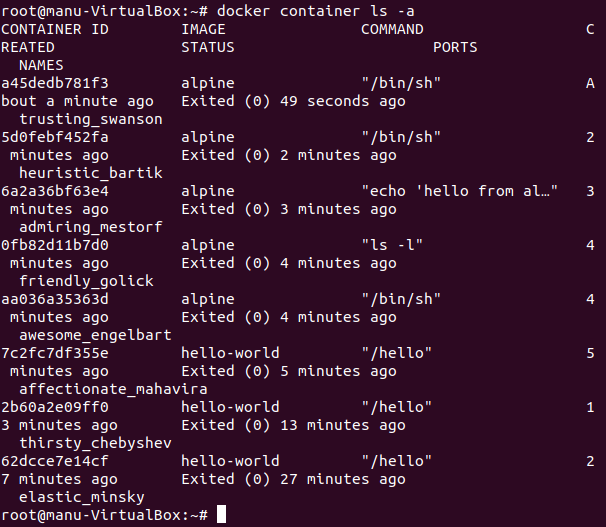


El “**docker container ls”** solo muestra todos los contenedores que se están ejecutando actualmente:



Como no hay contenedores en ejecución, verá una línea en blanco. Probemos una variante más útil:

**“docker container ls -a”.**

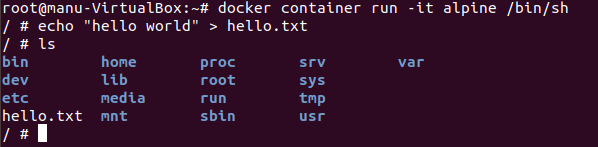


* Aislamiento del contenedor

En los pasos anteriores ejecutamos varios comandos a través de instancias de contenedor con la ayuda de “**Docker container run**”.

Escribimos el comando “**docker container run -it alpine /bin/ash”**

Una vez que se lanza el contenedor y se encuentra en el indicador de comandos del contenedor, escribimos el comando **echo "hello world" > hello.txt** y luego **ls**.



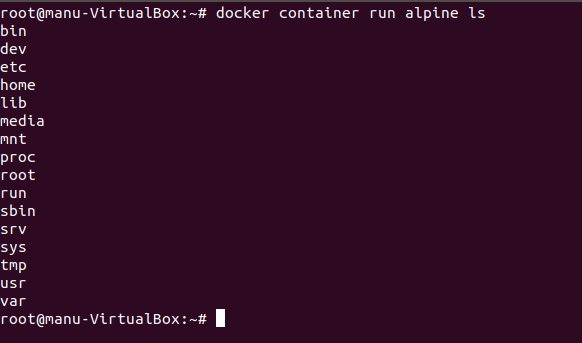
El primer echo crea un archivo llamado hello.txt con palabras “**hello world**” dentro del el.

El segundo comando le proporciona una lista de directorios de los archivos y debe mostrar el archivo “hello.txt” recién creado. Para salir tecleamos **exit**.

Para mostrar cómo funciona el aislamiento, ejecuta lo siguiente:

**“docker container run alpine ls”**

Es el mismo ls que usamos dentro del contenedor de cenizas interactivo del contenedor, pero esta vez, no existe cuyo archivo llamado “hello.txt” ¡Eso es aislamiento!

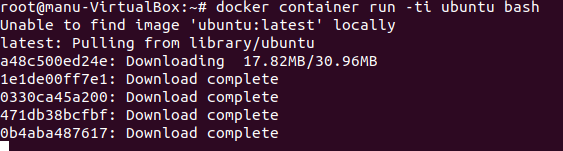


[Customizing Docker Images](https://training.play-with-docker.com/ops-s1-images): Move on to building your own custom Docker images.

Comencemos ejecutando un shell interactivo en un contenedor Ubuntu:

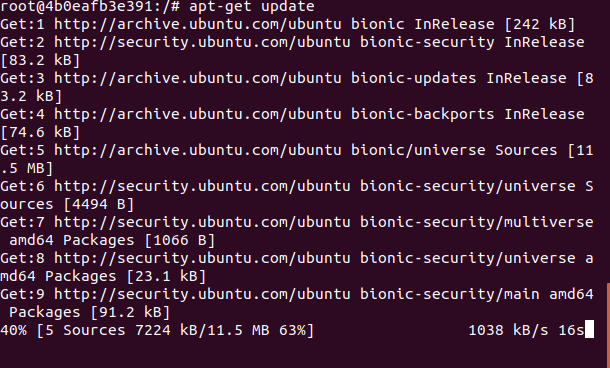
**“docker container run -ti ubuntu bash”**

**Como sabemos acabamos de tomar la imagen llamada “Ubuntu” de Docker Store y ahora están ejecutando el Shell bash dentro de ese contenedor.**

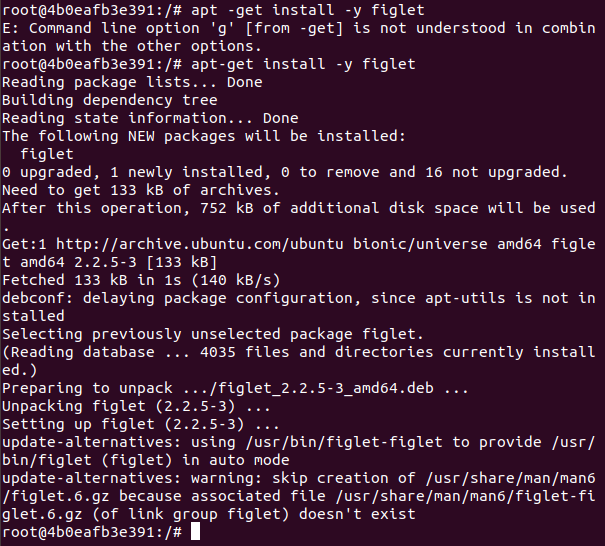


Para personalizar un poco las cosas, instalaremos un paquete llamado figlet en este contenedor. Su contenedor debería seguir ejecutándose, escriba los siguientes comandos en la línea de comandos de su contenedor ubuntu:

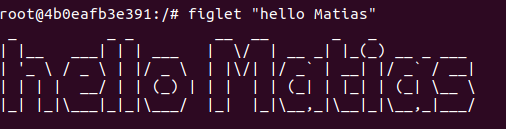
**“apt-get update”**



**“apt-get install -y figlet”**



**“figlet "hello Matias"**



[Deploy and Managing Multiple Containers](https://training.play-with-docker.com/ops-s1-swarm-intro): Real apps consist of multiple components.

* Inicializa tu enjambre

Lo primero que debemos hacer es decirles a nuestros anfitriones Docker que queremos utilizar el modo Docker Swarm.

Inicializar el modo Docker Swarm es fácil. En la primera ventana de terminal etiquetada, ingrese lo siguiente:

**“docker swarm init --advertise-addr $(hostname -i)”**

Eso es todo, ahora tiene su primer administrador de Swarm y está escuchando en la dirección IP devuelta.

