Docker Client – Recuperar Logs de salida

Hemos estado hablando de las diferencias entre iniciar o crear un contenedor. Hemos ejecutado por ejemplo el comando *docker create busybox echo hi there*, con el que obtuvimos como resultado la impresión del ID del contenedor:

```
jorge@MacBook-Pro-de-Jorge ~ % docker create busybox echo hi there
781a81c31b06f1087450c88d44b6b4d195b59b518f2b6f21877f8c8f090c205e
jorge@MacBook-Pro-de-Jorge ~ %
```

Después, aprendimos que podemos iniciar el contenedor recién creado usando el comando *docker start* seguido del ID del contenedor:

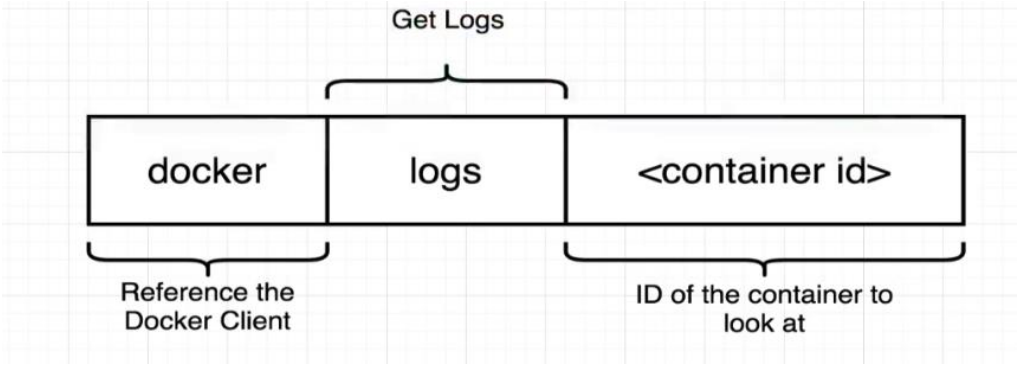
```
jorge@MacBook-Pro-de-Jorge ~ % docker start 781a81c31b06f1087450c88d44b6b4d195b59b518f2b6f21877f8c8f090c205e
781a81c31b06f1087450c88d44b6b4d195b59b518f2b6f21877f8c8f090c205e
jorge@MacBook-Pro-de-Jorge ~ %
```

Sin embargo, vimos un pequeño inconveniente con estos dos comandos, y es que para ver la información que se imprime desde el contenedor, tenemos que agregar el argumento *–a*.

Entonces, para obtener lo que el contenedor imprime, o para obtener todas sus salidas, ¿cómo podemos ejecutar el contenedor sin tener que agregar ese argumento?

Por ejemplo, imaginemos que tenemos un contenedor muy pesado, cuyo inicio es muy costoso y ejecutar *docker start* tarda varios minutos. S olvidamos agregar el argumento *–a*, tendríamos que detener el contenedor, volverlo a ejecutar con *docker start –a* y volver a esperar a que se inicie el contenedor.

Para evitar ese tipo de problemas, podemos utilizar un comando adicional llamado *docker logs* seguido del ID del contenedor del que queremos obtener sus salidas:



El comando logs nos permite mirar un contenedor y recuperar toda la información que se ha emitido desde él.

Vamos a probarlo. Primero creamos el contenedor con *docker create busybox echo hi there*:

```
jorge@MacBook-Pro-de-Jorge ~ % docker create busybox echo hi there
82d68a2801d36f299aac88abf257842ad0011c16f8cd6563a14d92a1584f50bb
jorge@MacBook-Pro-de-Jorge ~ %
```

Luego ejecutamos docker start seguido de del ID de nuestro contenedor:

```
jorge@MacBook-Pro-de-Jorge ~ % docker start 82d68a2801d36f299aac88abf257842ad0011c16f8cd6563a14d92a1584f50bb
82d68a2801d36f299aac88abf257842ad0011c16f8cd6563a14d92a1584f50bb
jorge@MacBook-Pro-de-Jorge ~ %
```

Con este comando se ejecuta nuestro contenedor, se imprime el mensaje *hi there* y se cierra inmediatamente.

Ahora, queremos volver a ese contenedor detenido y obtener todos los logs que se emitieron dentro de él, para esto hacemos un *docker logs* seguido del ID de ese contenedor:

```
jorge@MacBook-Pro-de-Jorge ~ % docker logs 82d68a2801d36f299aac88abf257842ad0011c16f8cd6563a14d92a1584f50bb
hi there
jorge@MacBook-Pro-de-Jorge ~ %
```

Veremos el resultado de los logs emitidos por ese contenedor, a pesar de ya estar detenido, que es simplemente el mensaje ”hi there”.

Una cosa importante que debe quedar clara es que al ejecutar el comando docker logs, de ninguna manera estamos ejecutando o reiniciando el contenedor, solamente estamos obteniendo los logs emitidos mientras se ejecutaba, los cuales se quedaron guardados dentro del contenedor.