Docker

Introducción:

Hoy en día existe una cantidad considerable de aplicaciones, mismas que en su tiempo fueron desarrolladas y puestas a prueba para su funcionamiento eficiente. Cuando se desarrolla software siempre existe la posibilidad de que se tengan complicaciones, ya sea en el ámbito de la construcción, distribución y ejecución del código. Docker viene a solucionar parte de estas complicaciones, hoy en día es una herramienta de gran ayuda es muy importante conocerlo.

Docker es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones. Permite separar sus aplicaciones de su infraestructura para que se pueda entregar software rápidamente.

Actualmente al aprovechar su metodología para enviar, probar e implementar código de manera rápida se puede reducir significativamente el retraso entre escribir código y ejecutarlo en producción. A continuación, te mostraremos parte de lo que es Docker como sus distintos conceptos y algunos ejemplos de su implementación.

. ¿Qué es Docker?

[Docker](https://www.docker.com/what-docker) es una plataforma que permite empaquetar software en unidades llamadas contenedores que incluyen todo lo necesario para que este se ejecute. Permite a los desarrolladores empaquetar aplicaciones en contenedores: componentes ejecutables estandarizados que combinan el código fuente de la aplicación con las bibliotecas del sistema operativo (SO) y las dependencias necesarias para ejecutar dicho código en cualquier entorno.

Imágenes Docker:

Las imágenes Docker son plantillas (que incluyen una aplicación, los binarios y las librerías necesarias) que se utilizan para construir contenedores Docker y ejecutarlos (los contenedores ejecutarán una imagen previamente compilada).

También podemos decir que las imágenes Docker son instancias de un contenedor.

Un contenedor se inicia ejecutando una imagen. El archivo *Dockerfile* define una imagen, y por tanto cómo será el contenedor que se creará a partir de esa imagen.

Y ese *Dockerfile* (que es un documento de texto) contiene todos los comandos que normalmente se ejecutan manualmente para construir una imagen para Docker. Docker puede leer las instrucciones del archivo *Dockerfile* y construir una imagen para Docker automáticamente.

1.- Pasos para crear una imagen Docker:

Vamos a ver un resumen de los pasos necesarios para crear una imagen en Docker.

–> 1. Creamos nuestro Dockerfile, teniendo en cuenta la [lista de instrucciones de los Dockerfile](https://docs.docker.com/engine/reference/builder/#environment-replacement):

* ADD
* COPY
* ENV
* EXPOSE
* FROM
* LABEL
* STOPSIGNAL
* USER
* VOLUME
* WORKDIR

--> 2. Después creamos la imagen con ***docker build*** , a partir del Dockerfile creado.

–> 3. Y finalmente la ejecutamos con ***docker run***.

2. Crear una imagen Docker (Ejemplo creando un “Hola Mundo“)

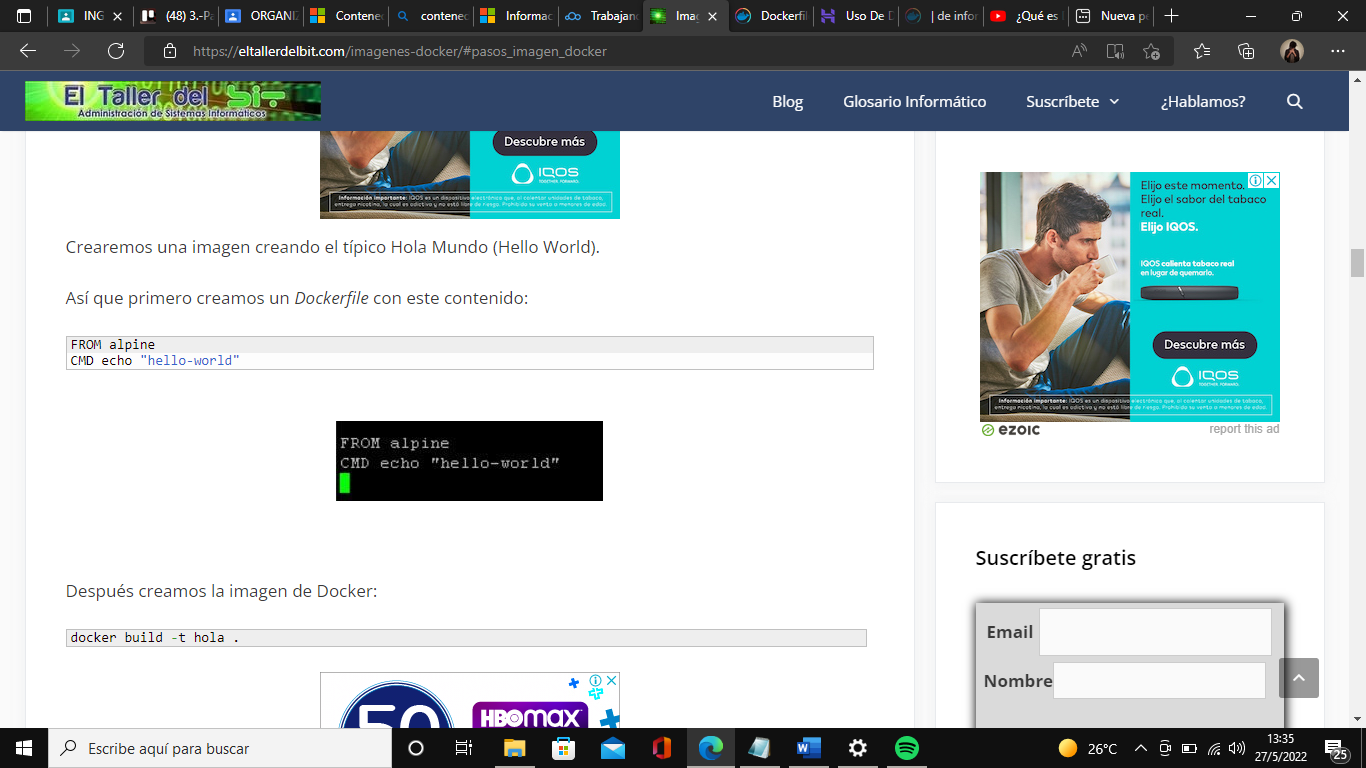
Las imágenes Docker se crean con un archivo llamado ***Dockerfile***.

Crearemos una imagen creando el típico Hola Mundo (Hello World).

Así que primero creamos un Dockerfile con este contenido:

FROM alpine

CMD echo "hello-world"

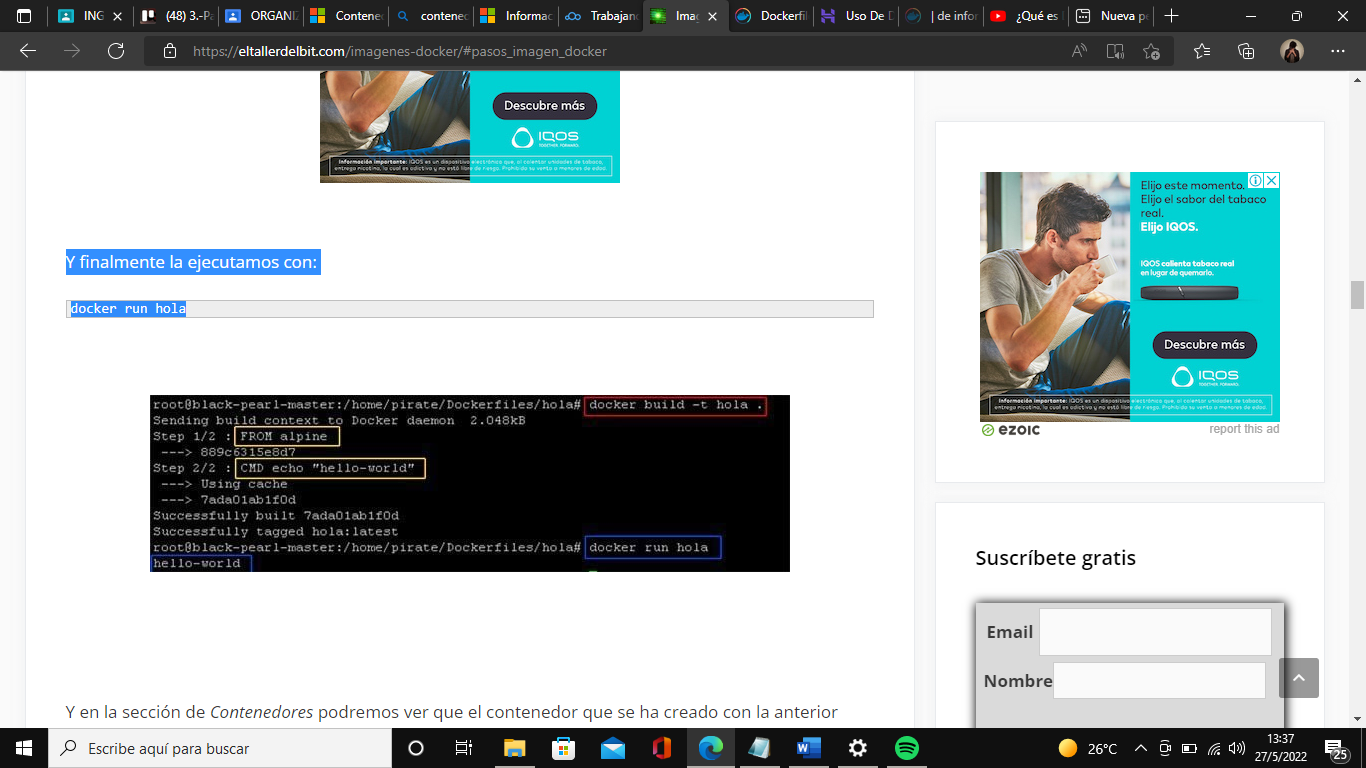


Después creamos la imagen de Docker:

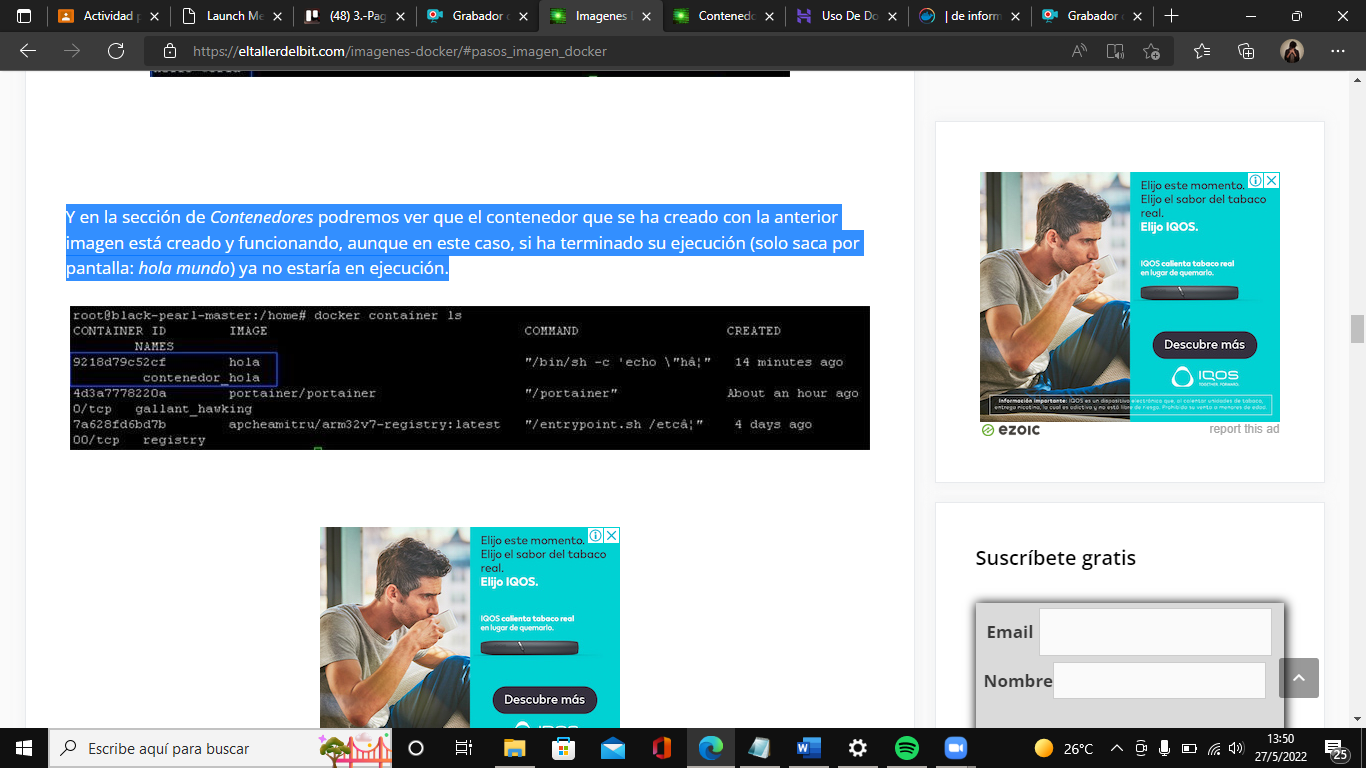
|  |
| --- |
| docker build -t hola . |

Y finalmente la ejecutamos con:

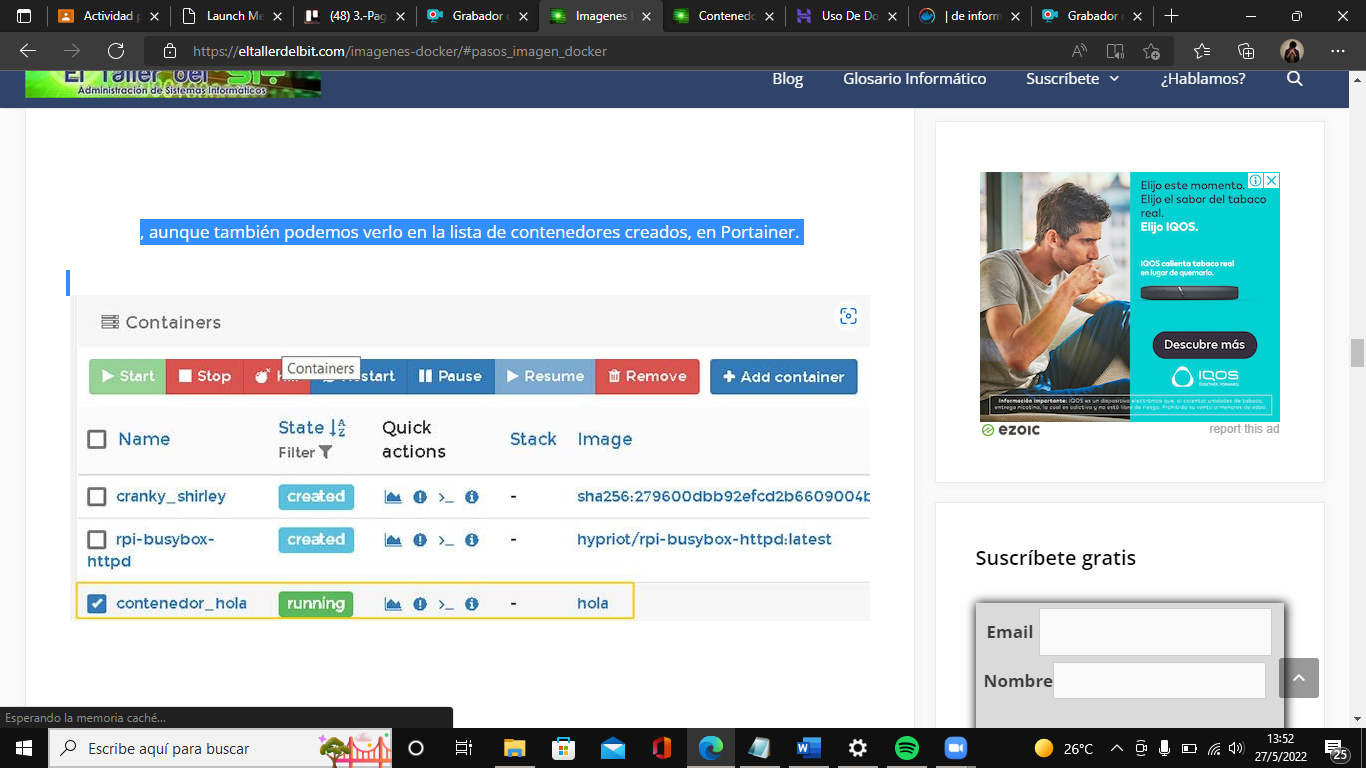
|  |
| --- |
| docker run hola |



Y en la sección de Contenedores podremos ver que el contenedor que se ha creado con la anterior imagen está creado y funcionando, aunque en este caso, si ha terminado su ejecución (solo saca por pantalla: hola mundo) ya no estaría en ejecución.



, aunque también podemos verlo en la lista de contenedores creados, en [Portainer](https://eltallerdelbit.com/docker-portainer/).



Sobre la imagen que hemos creado, podemos fijarnos en que la primera linea es:

|  |
| --- |
| FROM alpine |

La instrucción FROM inicializa una nueva etapa de compilación, y establece la imagen base para las instrucciones posteriores.

En este caso utiliza la imagen de ALPINE disponible en los repositorios públicos como los que hemos nombrado anteriormente.

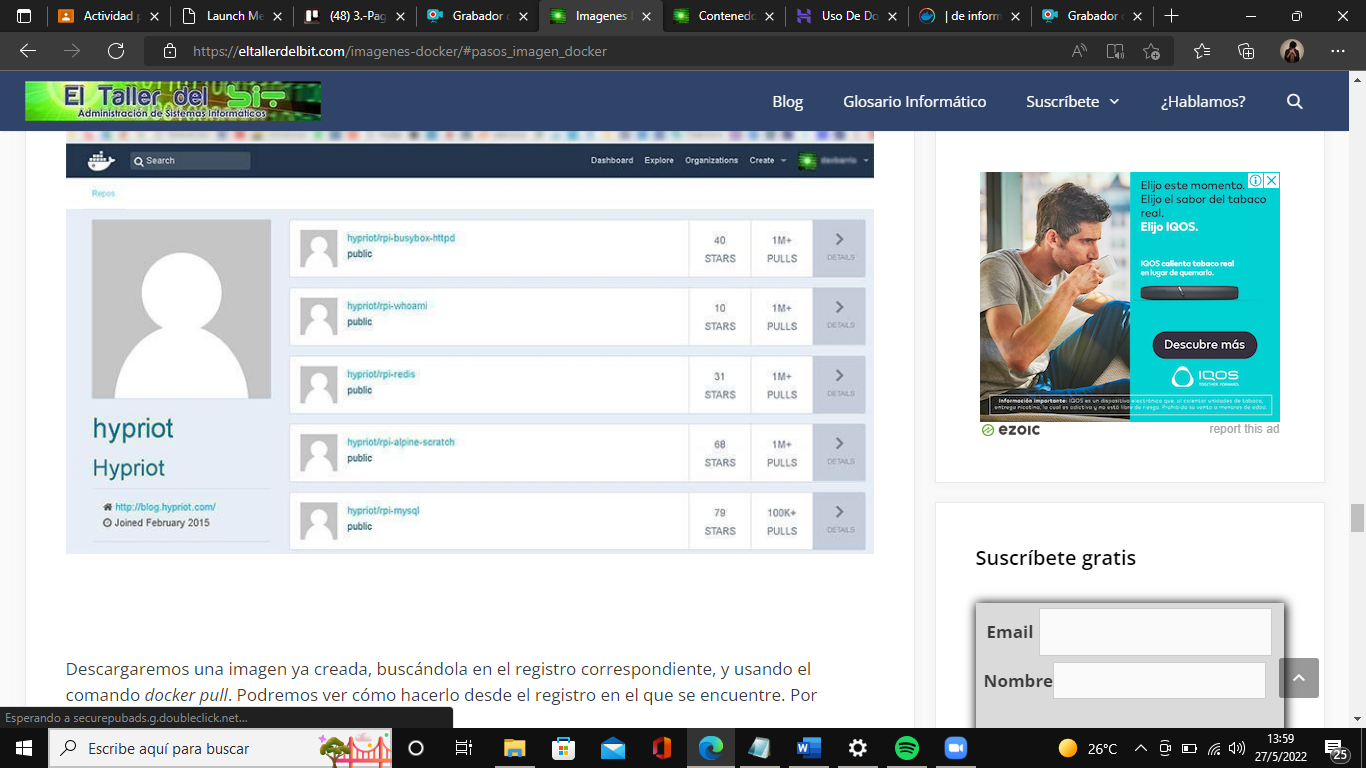
3. Descargar una imagen Docker ya creada.

 Si queremos utilizar una imagen ya creada, podemos hacerlo descargándola desde un registro a nuestra máquina Docker.

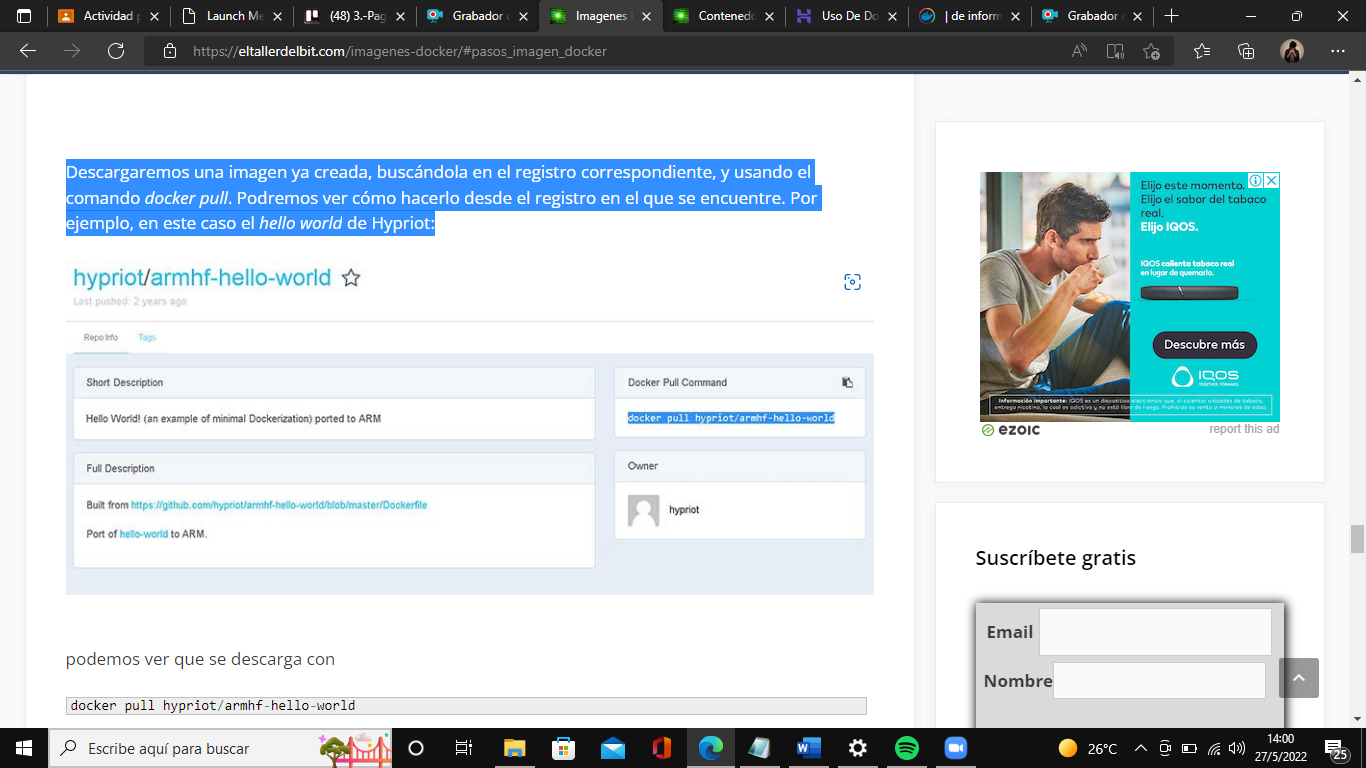
 Registros donde podemos encontrar imágenes Docker:

* [Docker Cloud](https://hub.docker.com/)
* [Docker Hub](https://hub.docker.com/)
* [Github Docker](https://github.com/docker)
* También podríamos crear nuestro propio registro local.

a continuación todas las [imagenes de Hypriot OS para Docker en Raspberry Pi (ARM)](https://hub.docker.com/u/hypriot/" \t "_blank):

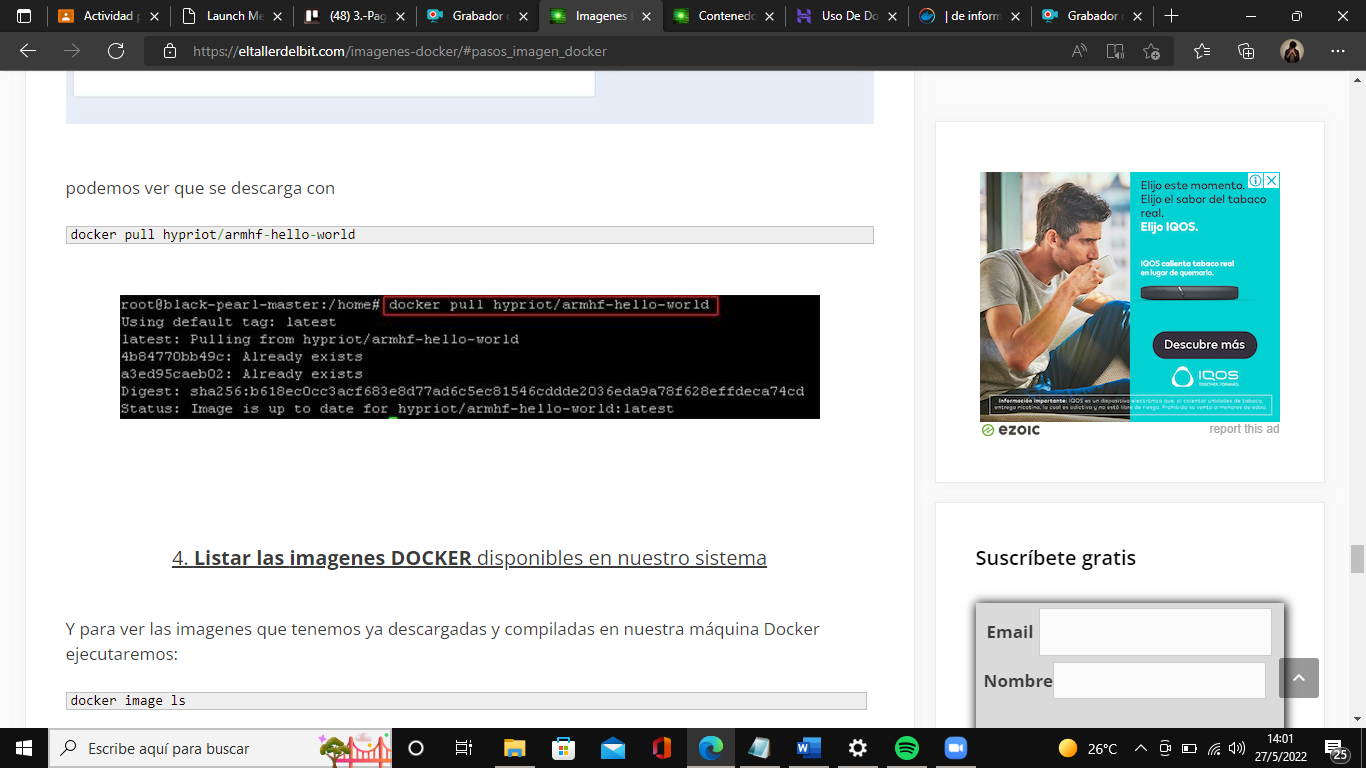


Descargaremos una imagen ya creada, buscándola en el registro correspondiente, y usando el comando docker pull. Podremos ver cómo hacerlo desde el registro en el que se encuentre. Por ejemplo, en este caso el [hello world de Hypriot](https://hub.docker.com/r/hypriot/armhf-hello-world/):



podemos ver que se descarga con:

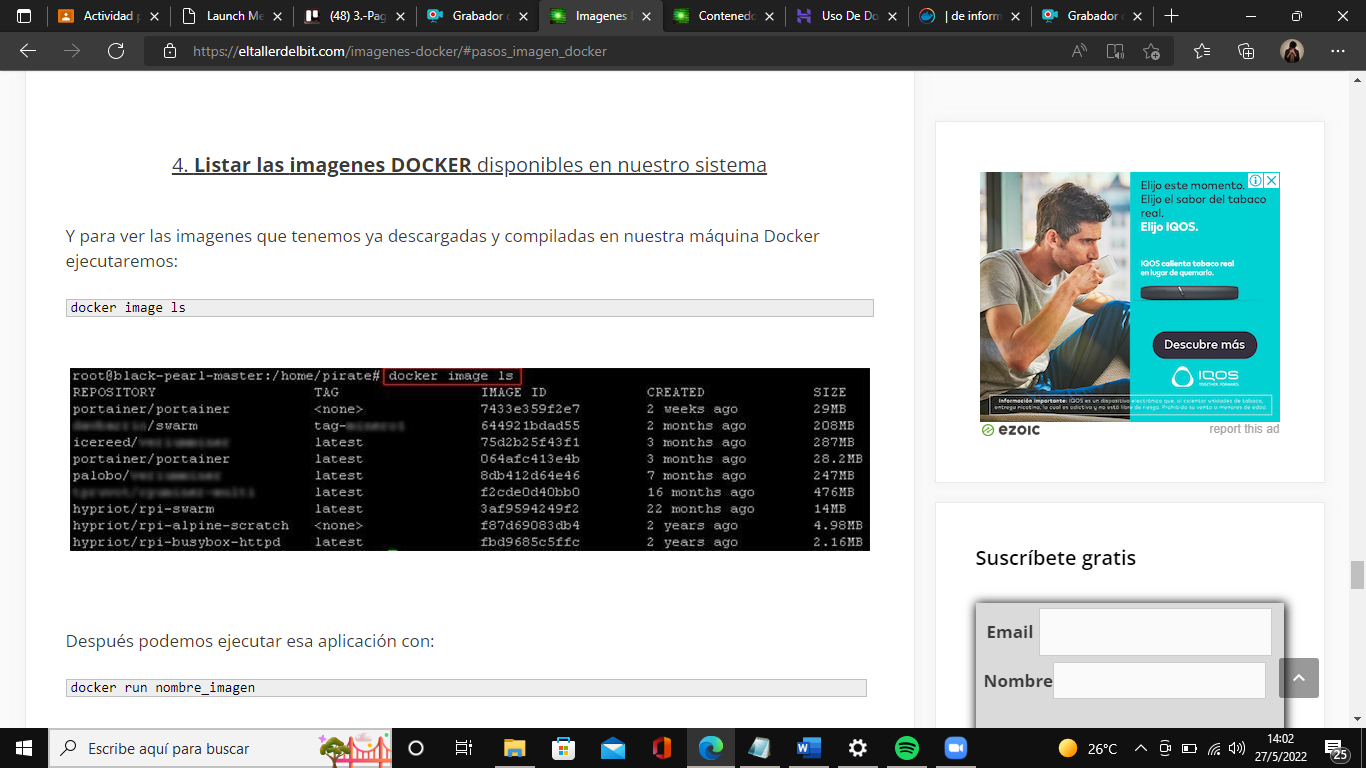
|  |
| --- |
| docker pull hypriot/armhf-hello-world |



4. Listar las imagenes DOCKER disponibles en nuestro sistema

 Y para ver las imagenes que tenemos ya descargadas y compiladas en nuestra máquina Docker ejecutaremos:

|  |
| --- |
| docker image ls |



Después podemos ejecutar esa aplicación con:

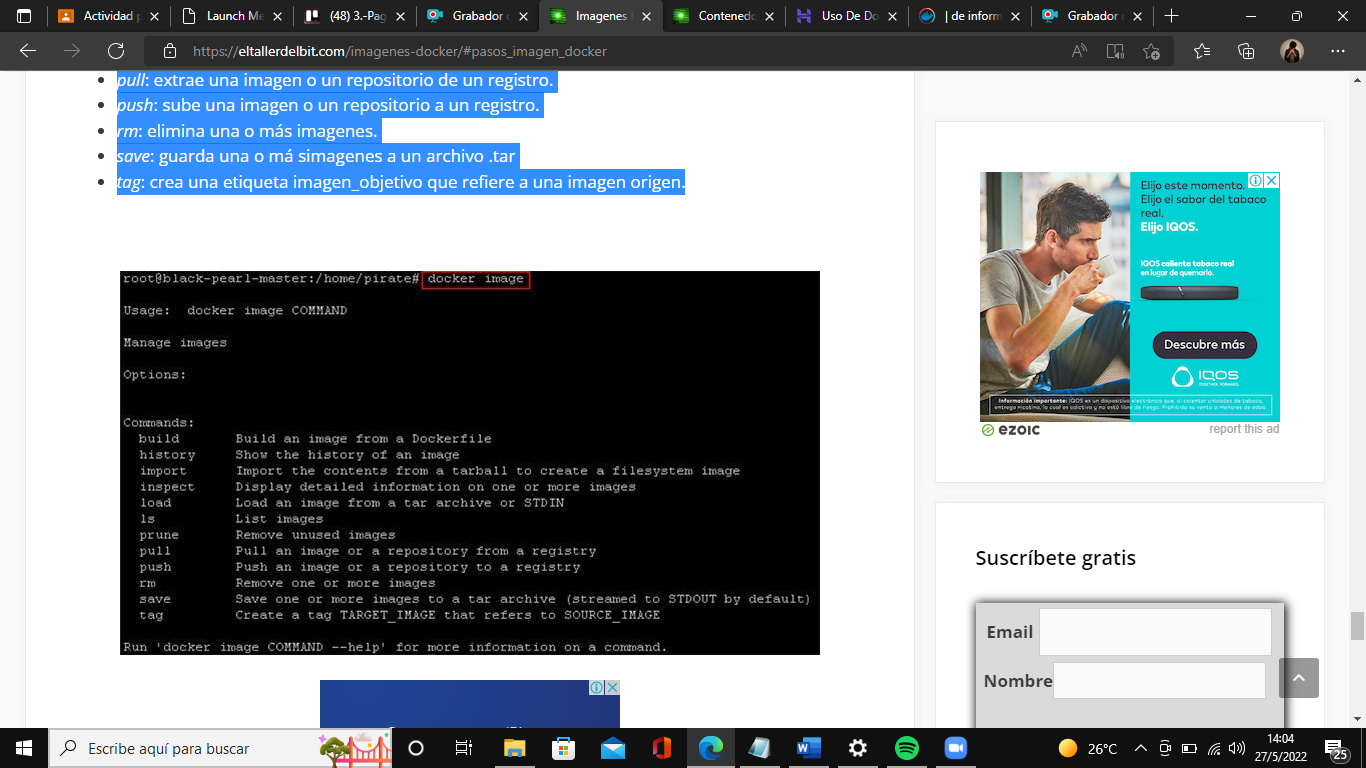
|  |
| --- |
| docker run nombre\_imagen |

Las imagenes podemos compartirlas por medio del registro.  
 Y las etiquetaremos con:

|  |
| --- |
| docker tag image username/repository:tag |

5. Otros comandos de ***docker image:***

* build: Construye una imagen a partir de un Dockerfile.
* history: muestra el historial de una imagen.
* import: importa el contenido de un comprimido para crear una imagen de sistema de archivos.
* inspect: muestra información detallada sobre una o más imagenes:
* load: carga una imagen desde un archivo .tar
* ls: lista las imagenes.
* prune: Desecha las imagenes no utilizadas.
* pull: extrae una imagen o un repositorio de un registro.
* push: sube una imagen o un repositorio a un registro.
* rm: elimina una o más imagenes.
* save: guarda una o má simagenes a un archivo .tar
* tag: crea una etiqueta imagen\_objetivo que refiere a una imagen origen.



Contenedores Docker:

[Docker](https://aws.amazon.com/es/docker/) es un proyecto que se basa en la contenedorización. Los contenedores Docker son pequeños sistemas encapsulados e independientes (tienen su propio árbol de procesos, sus interfaces de red.) y se pueden limitar los recursos que se asignan a cada contenedor.

Los contenedores Docker se ejecutan como un proceso, y consumen poca memoria. Están optimizados. El contenedor comparte el Kernel de la máquina host con otros contenedores.

Como funciona un contenedor Docker:

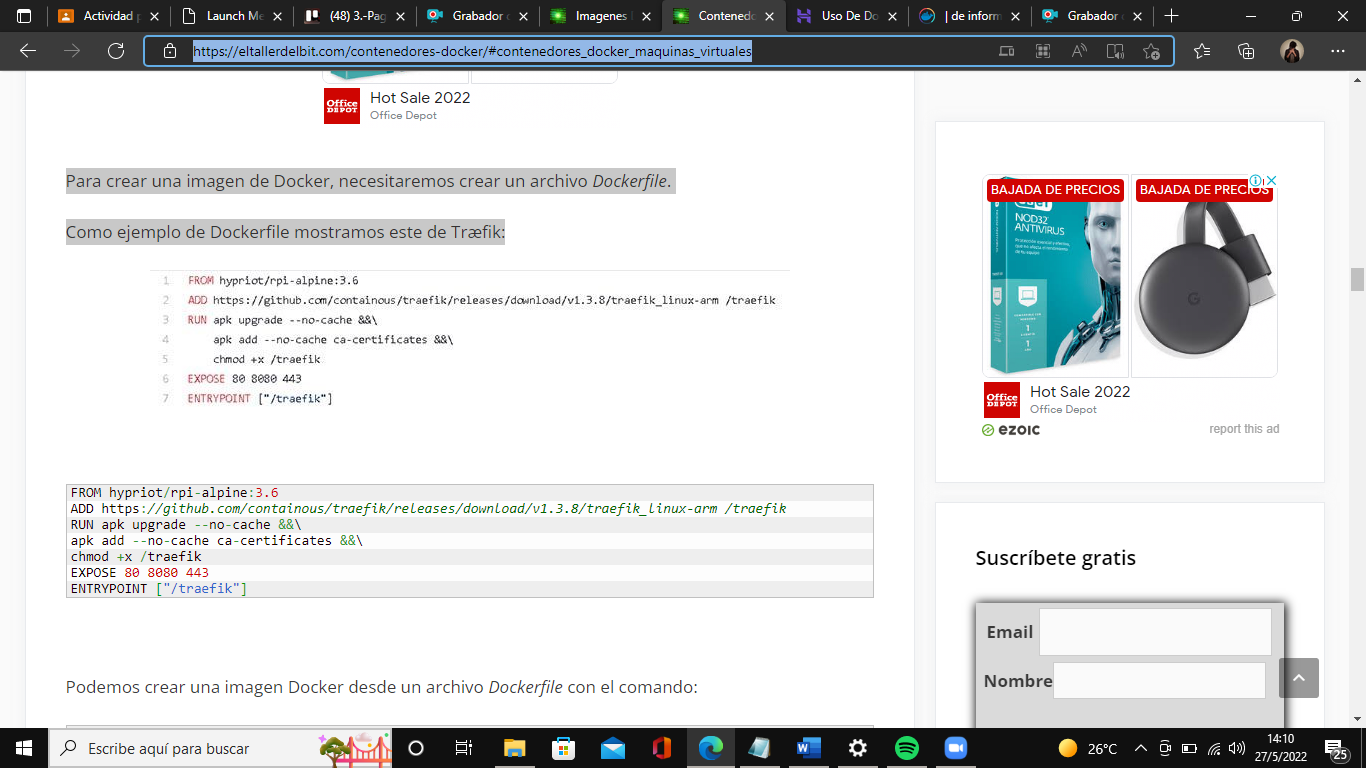
Un contenedor se inicia ejecutando una imagen. El archivo ***Dockerfile*** define una imagen, y por tanto cómo será el contenedor que se creará a partir de esa imagen. Por tanto, un contenedor es una instancia de una imagen.

Y ese ***Dockerfile*** (que es un documento de texto) contiene todos los comandos que normalmente se ejecutan manualmente para construir una imagen para Docker. Docker puede leer las instrucciones del archivo ***Dockerfile*** y construir una imagen para Docker automáticamente.

Crear una Imagen desde un ***Dockerfile***

Para [crear una imagen de Docker](https://eltallerdelbit.com/imagenes-docker), necesitaremos crear un archivo Dockerfile.

Como ejemplo de ***Dockerfile*** mostramos este de [Træfik](https://github.com/hypriot/rpi-traefik/blob/master/Dockerfile):



FROM hypriot/rpi-alpine:3.6

ADD https:*//github.com/containous/traefik/releases/download/v1.3.8/traefik\_linux-arm /traefik*

RUN apk upgrade --no-cache &&\

apk add --no-cache ca-certificates &&\

chmod +x /traefik

EXPOSE 80 8080 443

ENTRYPOINT ["/traefik"]

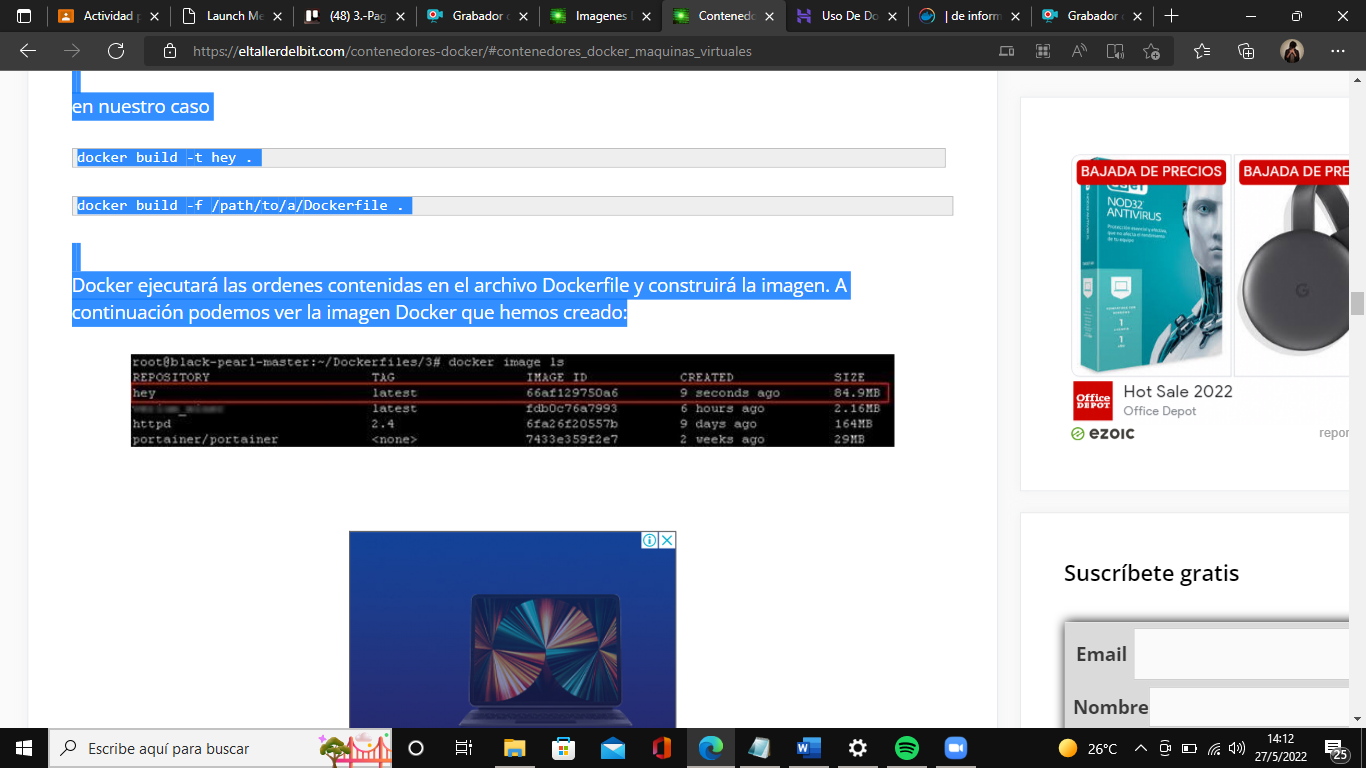
Podemos crear una imagen Docker desde un archivo Dockerfile con el comando:

|  |
| --- |
| docker build -t nombre\_imagen . |

en nuestro caso:

|  |
| --- |
| docker build -t hey . |
| docker build -f /path/to/a/Dockerfile . | |

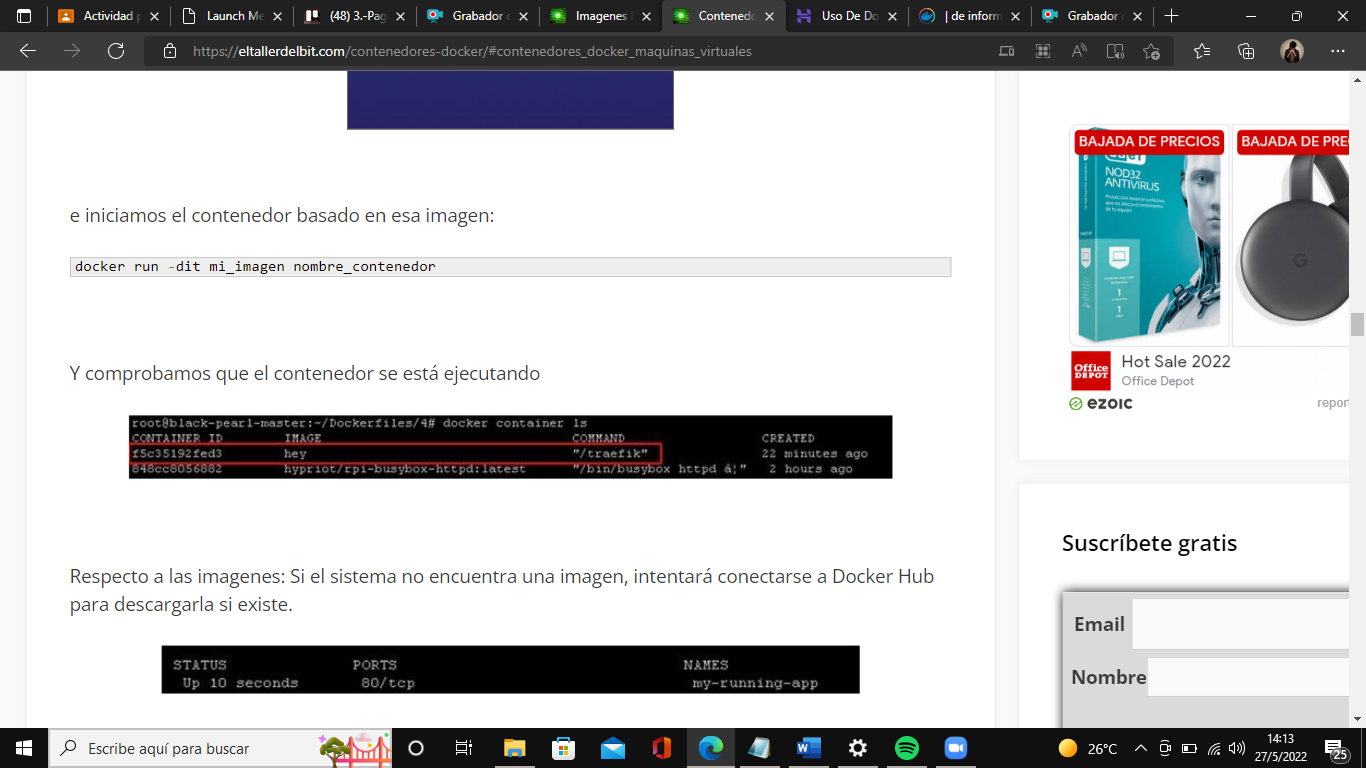
Docker ejecutará las ordenes contenidas en el archivo Dockerfile y construirá la imagen. A continuación, podemos ver la imagen Docker que hemos creado:



e iniciamos el contenedor basado en esa imagen:

|  |
| --- |
| docker run -dit mi\_imagen nombre\_contenedor |

Y comprobamos que el contenedor se está ejecutando:



Respecto a las imagenes: Si el sistema no encuentra una imagen, intentará conectarse a Docker Hub para descargarla si existe.

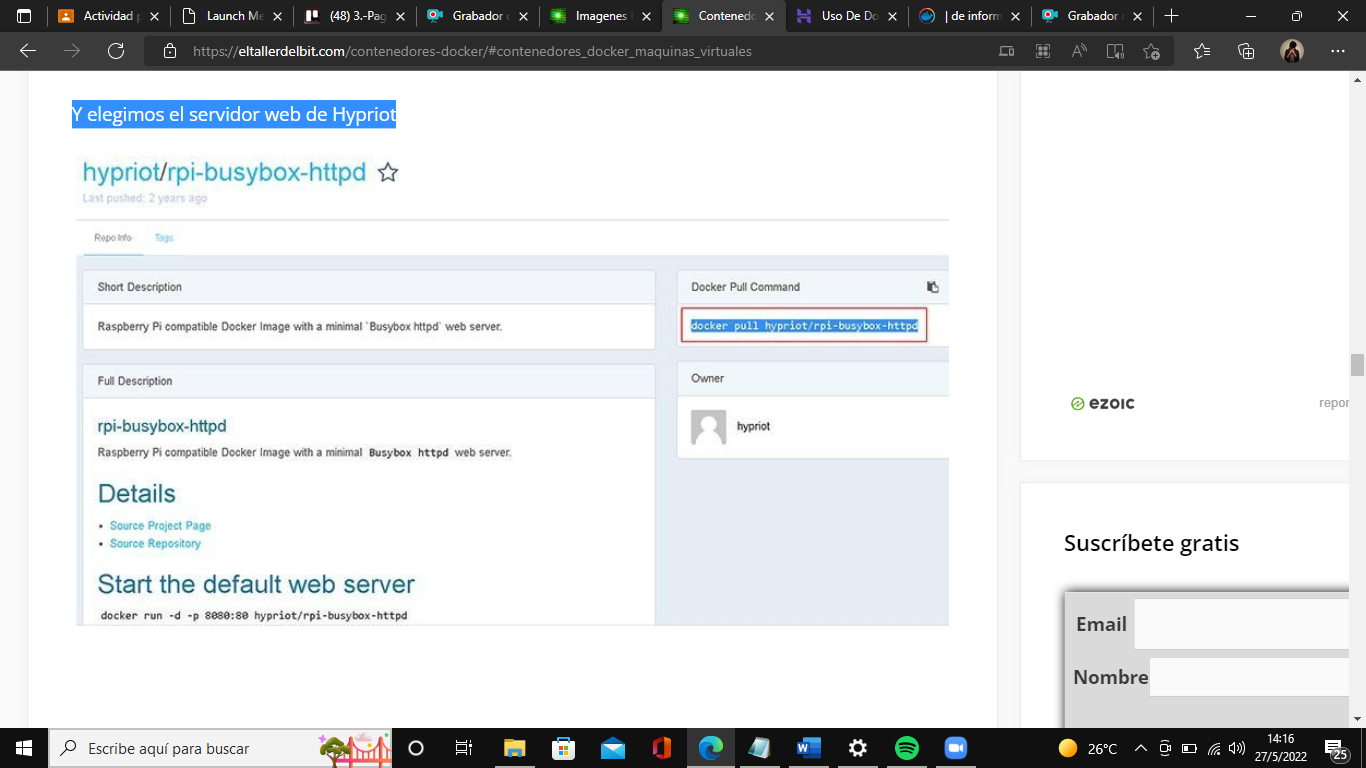


Ejemplo de creación de contenedor con una imagen ya creada (descargada de Docker Hub):

En este caso vamos a descargar una imagen ya compilada para poder crear un contenedor en una máquina con Docker.

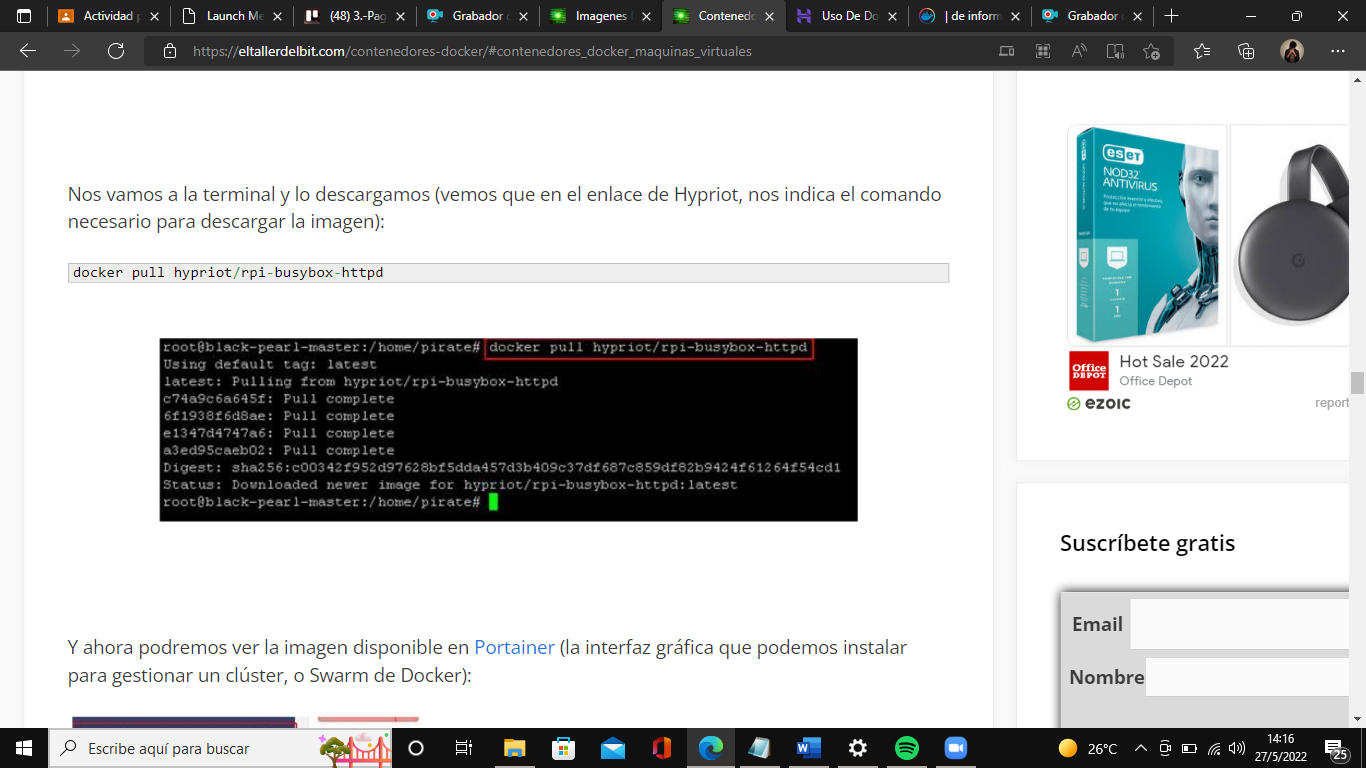
Recordemos que los contenedores que ejecutemos en [Raspberry Pi](https://eltallerdelbit.com/raspberry-pi/) han de provenir de imagenes compiladas para Raspberry (arquitectura ARM),  
   
podemos utilizar el repositorio de Hypriot en Docker Hub: <https://hub.docker.com/u/hypriot/>

Y elegimos [el servidor web de Hypriot](https://hub.docker.com/r/hypriot/rpi-busybox-httpd/)



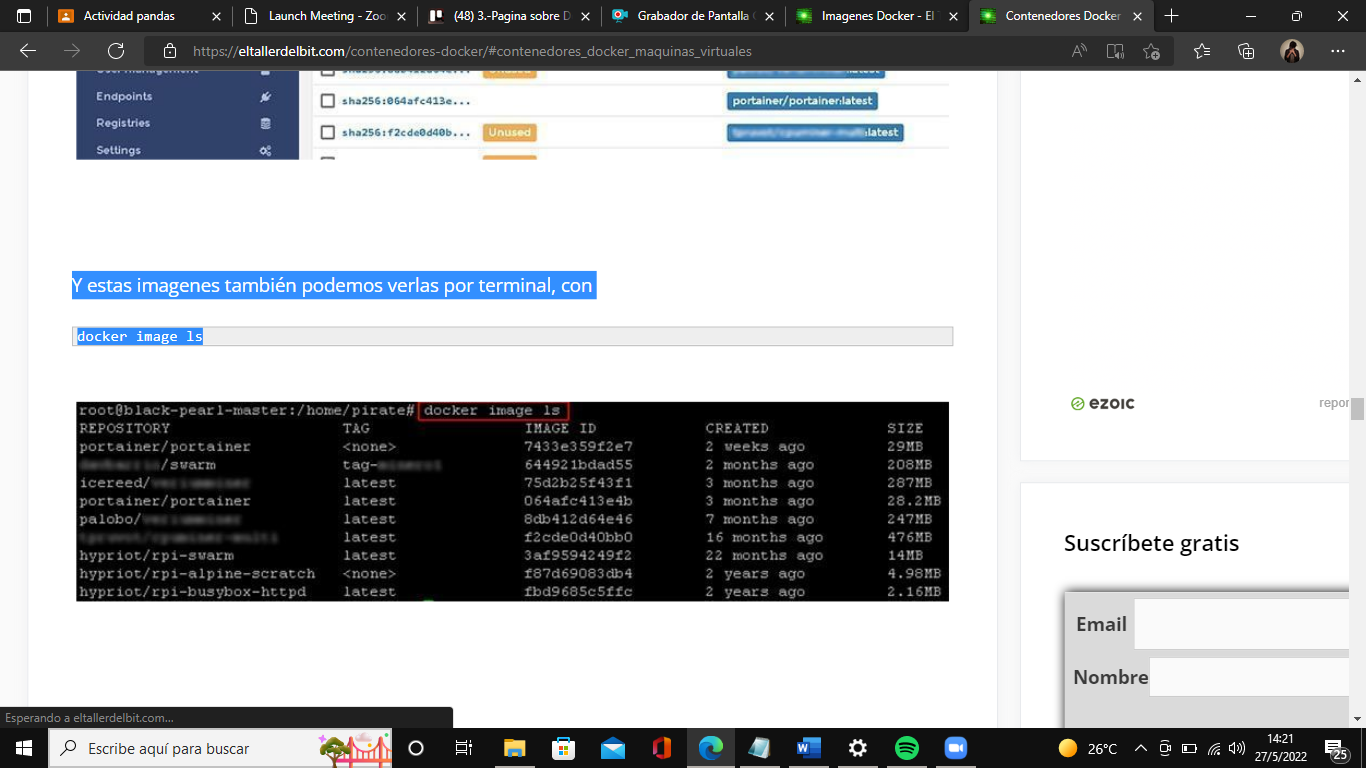
Nos vamos a la terminal y lo descargamos (vemos que en el enlace de Hypriot, nos indica el comando necesario para descargar la imagen):

|  |
| --- |
| docker pull hypriot/rpi-busybox-httpd |



Y estas imágenes también podemos verlas por terminal, con:

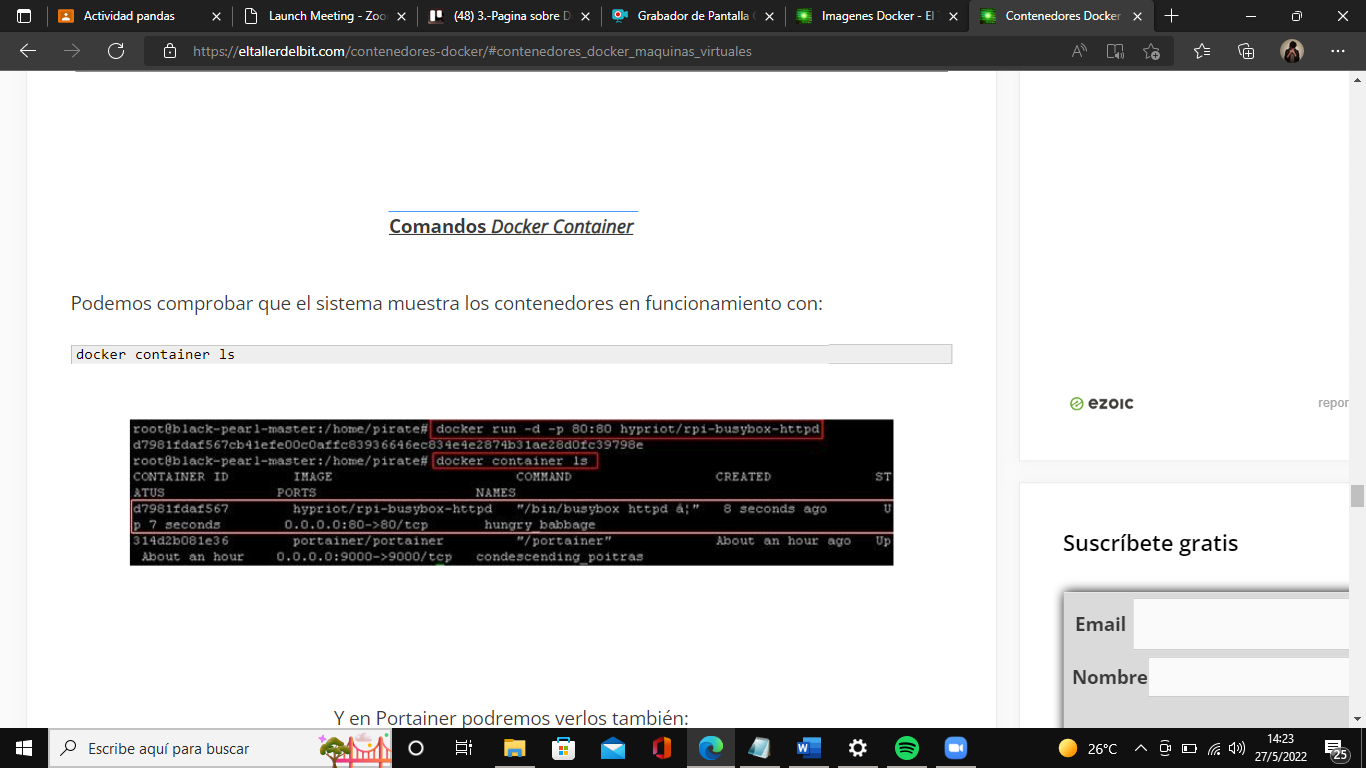
|  |
| --- |
| docker image ls |



Comandos Docker Container:

Podemos comprobar que el sistema muestra los contenedores en funcionamiento con:

|  |
| --- |
| docker container ls |

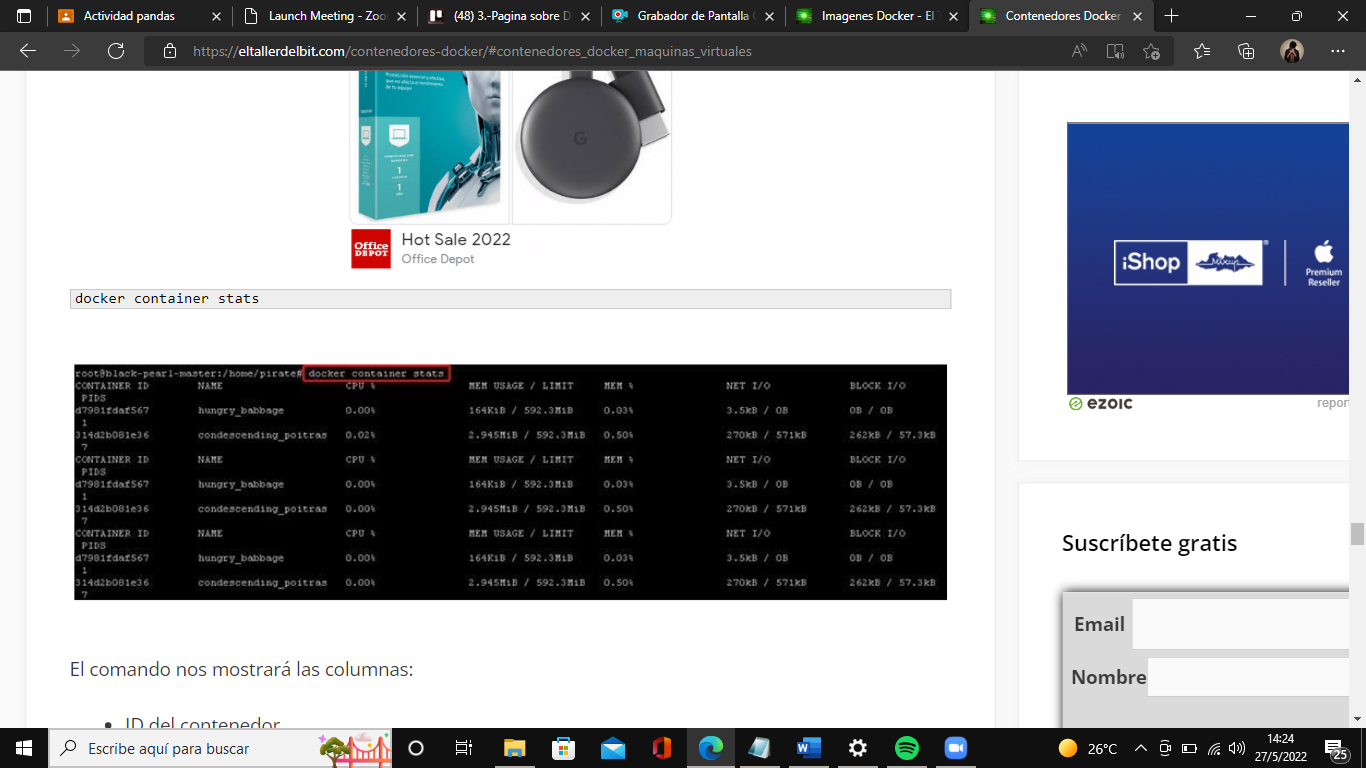


Se puede ver la lista de contenedores Docker en ejecución con el comando:

|  |
| --- |
| docker container ps |

Para ver el estado de los contenedores:

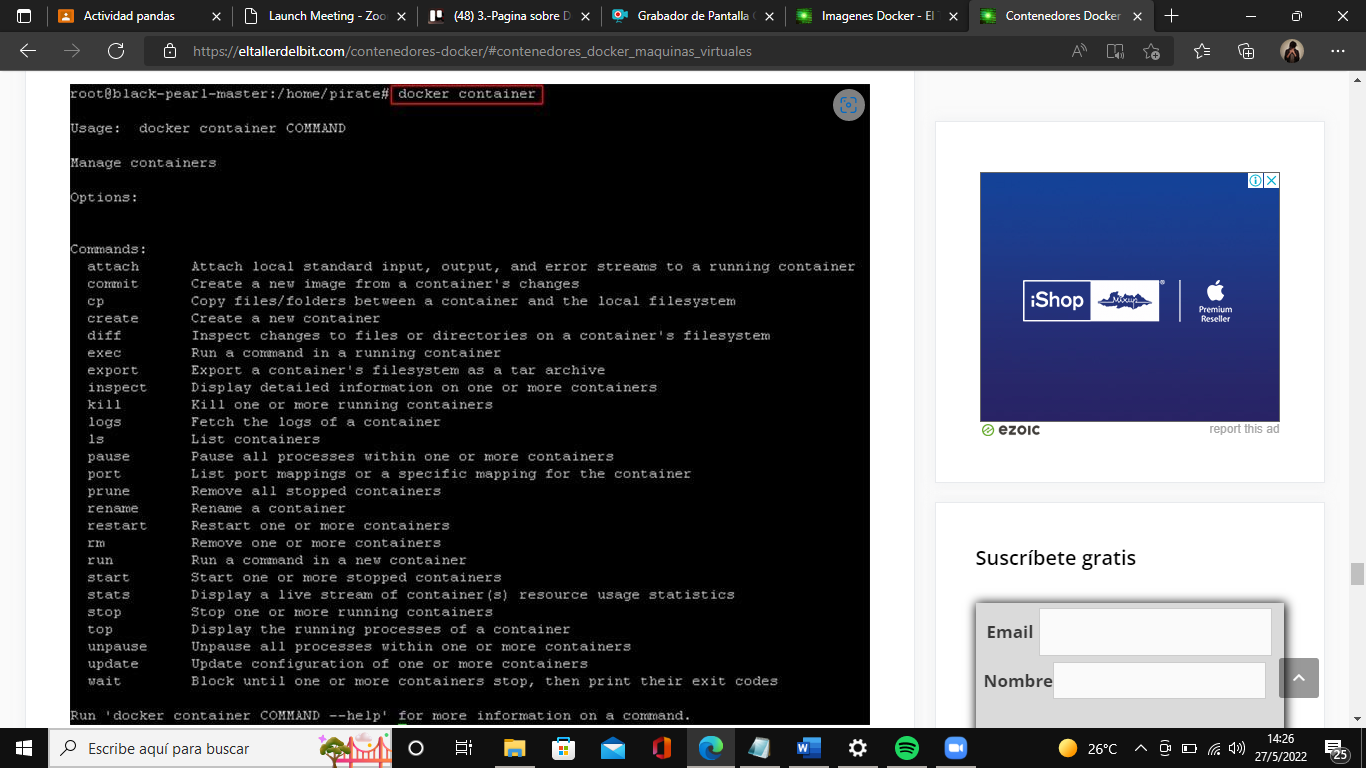
docker container stats



El comando nos mostrará las columnas:

* ID del contenedor
* Nombre.
* Porcentaje de CPU utilizado.
* Memoria utilizada/Límite de Memoria.
* Porcentaje de memoria utilizada.
* NET I/O (el tráfico de red).
* BLOCK IO (las lecturas y escrituras físicas en disco).

Otros comandos de Docker Container:



Links de consulta:

https://eltallerdelbit.com/imagenes-docker/

https://docs.microsoft.com/es-es/dotnet/architecture/microservices/container-docker-introduction/docker-containers-images-registries

https://eltallerdelbit.com/contenedores-docker/#contenedores\_docker\_maquinas\_virtuales