Contenedores y Docker

Sistemas web de altas prestaciones
2017-2018

Antonio Carrasco Castro
Fernando Roldán Zafra

1. Resumen

Si nos encontramos desarrollando una aplicación y queremos probar una versión concreta para la que se necesita una configuración concreta podemos encontrarnos con el problema de que debemos configurar nuestro equipo debidamente para poder realizar esta tarea. Para ello existen los contenedores y el proyecto Docker.
En este texto se van a presentar las principales características de los contenedores, como desplegarlos y además a como interactuar con Docker Cloud. La estructura básica que seguirá este trabajo es la siguiente:
-Definición de contenedor.
-Conceptos básicos.
-Contenedores de Windows 2016.
-Implementación y utilidades de los contenedores.

2. Introducción

Logicamente se pueden crear contenedores en cualquier SO, pero en este trabajo nos centraremos en los contenedores de Windows. Antes de empezar a hablar de estos contenedores de Windows o “Windows Containers” seria lógico definir lo que es un “contenedor” y también habría que definir el principio en el que se basa, la “virtualización”.

Podemos definir la virtualización como un proceso a través del cual se crea una representación software en lugar de realizarla sobre hardware.
A través de este proceso se pueden virtualizar servidores, redes, etc.
Una de las formas de virtualizar un sistema pasa por utilizar maquinas virtuales, las cuales emulan una nueva computadora dentro de la ya existente, simulando el hardware que utilizara la maquina. De esta manera, se puede instalar un sistema operativo dentro de otro.

Otro de los métodos de aplicar la virtualización es el uso de contenedores.
Un contenedor es un tipo de maquina virtual, la cual se diferencia de las maquinas virtuales convencionales en que los contenedores comparten una mayor cantidad de recursos con la máquina en la que corren.
Tal y como se puede leer de manos del director de tecnología de Azure Mark Russinovich en [1]: “*Por razones de eficiencia, muchos de los archivos del sistema operativo, directorios y servicios en ejecución se comparten entre contenedores y se proyectan al espacio de nombres de cada contenedor*”. De esta manera se consigue que los contenedores sean mas ligeros y que en una misma maquina puedan ejecutarse varios contenedores de forma eficiente.

No se puede hablar de contenedores sin hablar del proyecto “Docker”. Los contenedores se han venido utilizando en sistemas Unix desde hace bastante tiempo, pero con el lanzamiento del proyecto “Docker” como código abierto se incremento el uso de los contenedores enormemente.

El objetivo de este proyecto es crear contenedores ligeros y portables para aplicaciones, que puedan ejecutarse en cualquier maquina con Docker instalado independientemente del sistema operativo que haya debajo.

Una vez llegados a este punto estamos listos para abordar los contenedores de windows y hablar de sus peculiaridades.

3. Conceptos básicos

Caben destacar varios conceptos básicos de los contenedores [3]:
- Host del contenedor: Se trata de la maquina en la que esta instalada la característica de contenedores de Windows. Si el contenedor estuviera instalado dentro de una máquina virtual, el Host del contenedor sería la máquina virtual.

-Imagen del contenedor: Puede darse el caso de que queramos almacenar el estado de un contenedor en un momento dado y mantener los cambios que se hayan realizado en un

registro o en los sistemas de archivos del contenedor. Pues precisamente, la imagen del contenedor es eso, se trata de capturar el estado de un contenedor de forma que a través de la imagen se puedan crear nuevos contenedores que hereden los cambios que se hayan podido realizar.

-Espacio aislado: Espacio en el que trabaja el contenedor. Es “aislado” ya que todas las operaciones de escritura/lectura solo pueden realizarse en este espacio.

-Repositorio de contenedor: Al crear una imagen de contenedor esta y todos los datos necesarios se almacenan en un repositorio de contenedor. Esta imagen puede ser usada tantas veces como sea requerida por el Host del contenedor aunque también se pueden almacenar en repositorios públicos de manera que se puedan usar en varios Host de contenedor diferente.

-Imagen del sistema operativo de contenedor: Los contenedores están implementados a partir de capas de imágenes, y la primera de estas capas es la del sistema operativo.

Todos estos conceptos se pueden apreciar en la Figura 1.

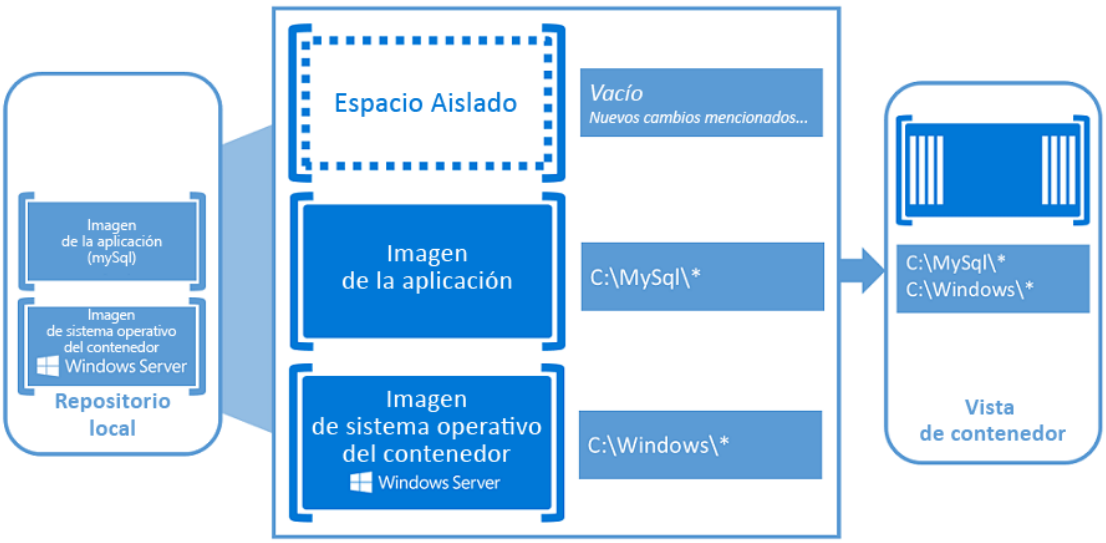


Figura 1: Vista de un contenedor

4. Contenedores de Windows Server 2016

Los sistemas operativos Windows nunca nos han ofrecido la posibilidad de trabajar con contenedores, pero con el lanzamiento de Windows Server 2016 esto cambió, se añadieron dos tipos de contenedores:

-Contenedores de Windows Server: Se trata de contenedores habituales, es decir, aíslan las aplicaciones que corren dentro de ellos y se caracterizan por que comparten el kernel con el Host del contenedor, el cual también es compartido por los demás contenedores de este tipo.

-Contenedores de Hyper-V: Este tipo de contenedores va un paso mas allá en lo que al aislamiento se refiere ya que estos se ejecutan en una maquina virtual altamente optimizada. Es decir, en este tipo de contenedores no se comparte el kernel del Host con los contenedores de Hyper-V.

Pero de estos dos tipos de contenedores puede surgir una duda, ¿En que contexto es mas útil un contenedor Hyper-V? La respuesta es muy sencilla, la razón principal para utilizar este tipo de contenedores es el aislamiento. Con este tipo, podríamos decir que no hay manera de escapar del contenedor por lo que las aplicaciones están aisladas de la máquina Host.

También hay que añadir que Windows permite el uso de la API de Docker en Windows.

5. Creación de un contenedor en Windows Server 2016

[4] Primero necesitamos un equipo físico o virtual con Windows Server 2016 con todas las actualizaciones criticas instaladas. Una vez tenemos el equipo preparado pasamos al siguiente paso:

-Instalación de Docker: Primero abrimos el PowerShell desde la consola usando el comando "powershell".
Para poder instalar Docker necesitamos instalar el proveedor de PackageManagement de Docker-Microsoft desde la galería de PowerShell con el siguiente comando:

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
```

Instalamos la ultima versión de Docker:

```
Install-Package -Name docker -ProviderName DockerMsftProvider
```

El PowerShell nos preguntara si confiamos en el origen del paquete ya que este, no esta marcado como de confianza. Marcamos "S" y seguimos con la instalación tal y como se puede ver en la figura 2.

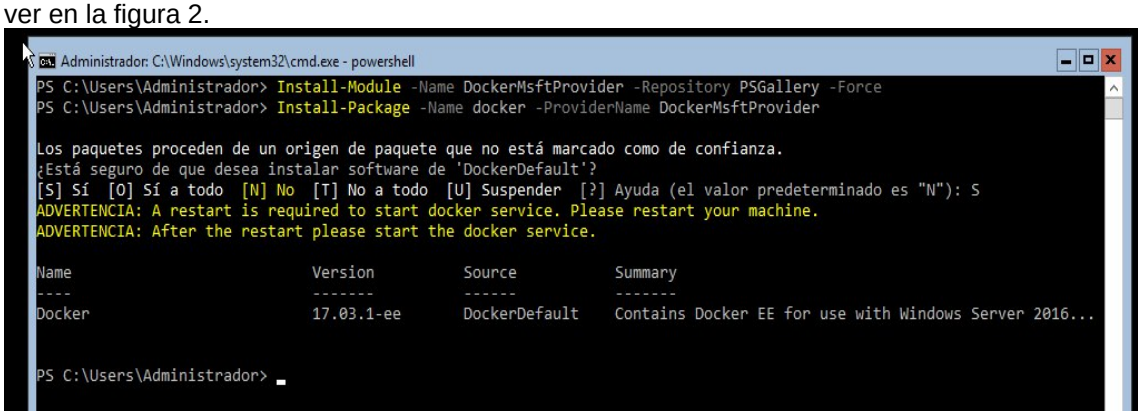


Figura 2: Instalación de Docker

Cuando el proceso termine reiniciamos el equipo con el comando:

```
Restart-Computer -Force
```

-Instalar actualizaciones de Windows: Para actualizar el sistema debemos acceder al menú de configuración de Windows Server, para ello utilizamos el siguiente comando:

```
sconfig
```

Una vez accedemos al menu que aparece en la figura 3, utilizamos la opción numero 6 para actualizar el sistema y mas tarde pulsamos “A” para seleccionar todas las actualizaciones.

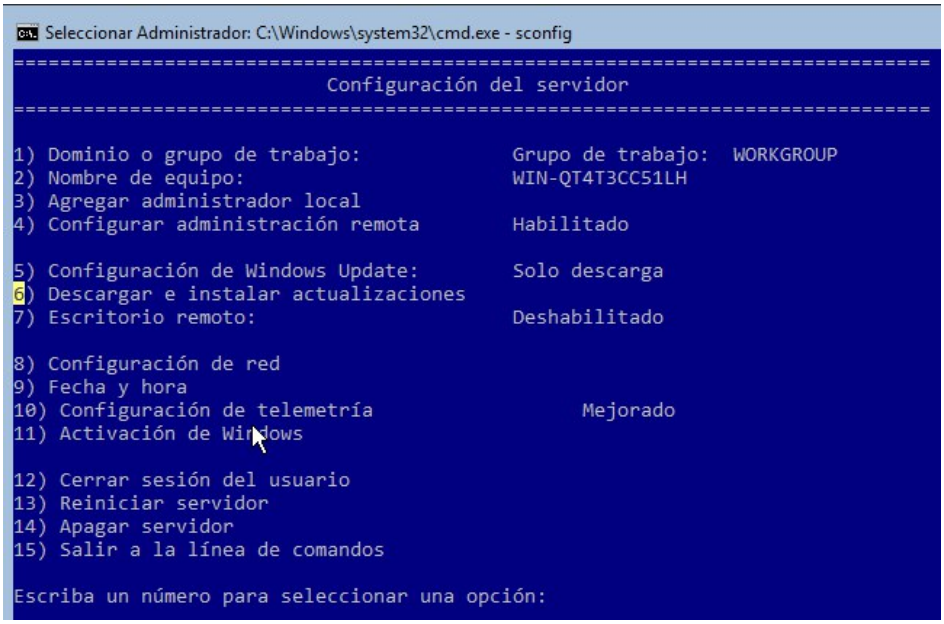
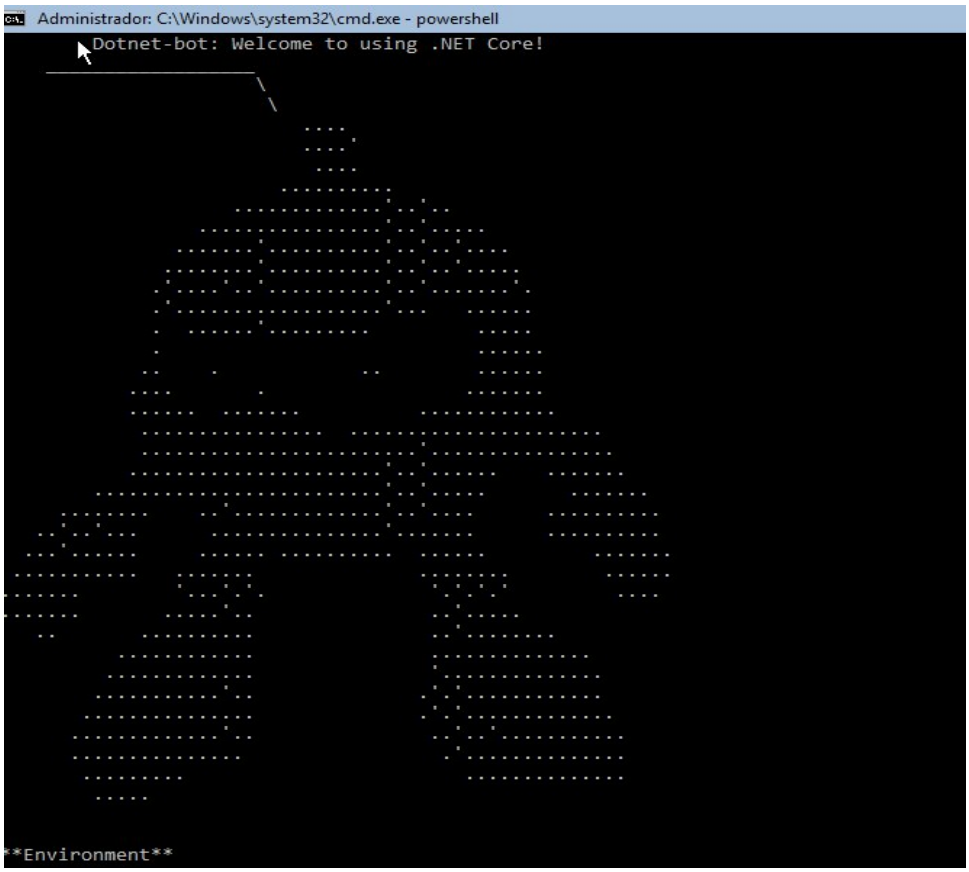


Figura 3: menú de configuración del servidor

-Implementación del primer container: Llegados a este punto ya estamos preparados para implementar el primer container, para ello descargaremos una imagen de ejemplo de .NET Core y ejecutaremos un contenedor simple que ejecuta una aplicación Hello world de .NET. Para ello usaremos el siguiente comando y obtenemos el resultado de la figura 4:

```
docker run microsoft/dotnet-samples:dotnetapp-nanoserver
```



6. Imágenes de contenedores

En el apartado anterior pudimos ver como iniciar un contenedor a partir de una imagen. Pero ¿Y si queremos implementar nosotros una imagen que podamos compartir con cualquier persona para que pueda ejecutarla en cualquier maquina con Docker? Hay varias formas de hacerlo, de forma manual o de forma automática.
-Forma manual: Primero debemos implementar un contenedor. En este caso implementaremos un contenedor de IIS (Internet Information Services)[5] a partir de una imagen de IIS creada previamente. Una vez hecho esto trabajaremos dentro de una shell dentro del contenedor:

```
docker run -d --name MiIIS -p 80:80 microsoft/iis
```

A continuación ejecutamos un cmd interactivo en el contenedor como podemos ver en la figura 5. Lo que nos permite ejecutar comandos sobre el contenedor en ejecución sin que se detenga el IIS o el contenedor:

```
docker exec -i MiIIS cmd
```

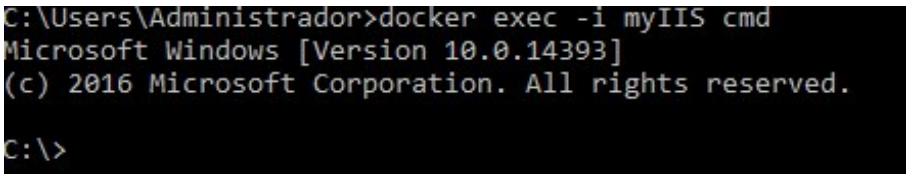


Figura 5: Shell Interactivo dentro del contenedor

En este punto podemos hacer un cambio en el contenedor en ejecución en este caso sera eliminar la pagina de presentación por defecto de IIS.

```
del C:\inetpub\wwwroot\iisstart.htm
```

Habiendo eliminado la pagina de presentación de IIS, esta claro que debemos especificar cual sera la nueva pagina, en este caso crearemos una nueva en la que solo se muestre un Hello World:

```
echo "Hello World! This is a windows server container :D" > C:\inetpub\wwwroot\index.html
```

Una vez hecho esto si accedemos a la ip del host del contenedor desde el navegador de otro equipo podemos ver que se muestra el mensaje que hemos introducido como podemos ver en la figura 6.

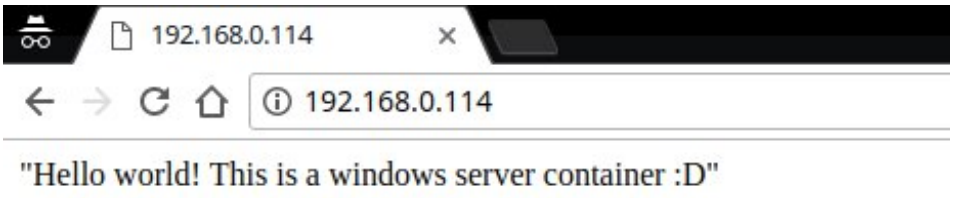


Figura 6: Contenedor modificado manualmente

En este punto podemos crear una imagen del contenedor modificado pero antes debemos conocer el nombre del contenedor ejecutando el comando, tal y como podemos ver en la figura 7:

```
docker ps -a
```

```
C:\>docker ps -a
CONTAINER ID        IMAGE                                          COMMAND                  CREATED             STATUS
                            PORTS               NAMES
fc1aa5e2f9c9        microsoft/iis                                  "C:\\ServiceMonitor..."   About an hour ago   Exited
 (1067) 55 minutes ago                          MiIIS
dd27a1672a3b        microsoft/dotnet-samples:dotnetapp-nanoserver  "dotnet dotnetapp.dll"    About an hour ago   Exited
 (0) About an hour ago                          friendly_pare
c05b8520bd56        microsoft/iis                                  "C:\\ServiceMonitor..."   2 hours ago         Exited
 (2147483648) About an hour ago   0.0.0.0:80->80/tcp   myIIS
ac53478f0afe        microsoft/dotnet-samples:dotnetapp-nanoserver  "dotnet dotnetapp.dll"    3 hours ago         Exited
 (0) 3 hours ago                                clever_noyce

C:\>
```

Figura 7: Lista de contenedores.

Ahora debemos detener el contenedor y hacer un commit para crear una imagen.

```
docker stop MiIIS
```

```
docker commit MiIIS iismodificado
```

Nota: A la hora de crear la imagen, el nombre que tendrá no debe contener mayúsculas.

Si queremos comprobar si se ha creado la imagen usaremos el siguiente comando como podemos ver en la figura 8.

```
docker images
```

```
PS C:\Users\Administrador> docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
iismodificado               latest              6d070a070c8e        5 minutes ago       10.7 GB
microsoft/dotnet-samples    dotnetapp-nanoserver 6313d7c84df3       14 hours ago        1.13 GB
microsoft/iis               latest              b02b292ce005        3 weeks ago         10.5 GB
PS C:\Users\Administrador>
```

Figura 8: Lista de imágenes de contenedores

Como vemos en la imagen anterior, se puede observar que se ha creado la imagen del contenedor “iismodificado”. Esta imagen ahora puede ser implementada y el contenedor resultante tendrá todos los cambios realizados.

-Forma automática: Este método permite automatizar la obtención de una imagen de un contenedor pero requerirá que tengamos un identificador de Dockerfile[6] que podemos obtener registrándonos en Docker cloud.

El primer paso es crear un directorio en C llamado build. Podemos hacerlo desde la powershell usando el comando:

```
New-Item -ItemType Directory -Path C:\build
```

De la misma manera creamos un fichero llamado Dockerfile sin ninguna extensión dentro del directorio que acabamos de crear y lo abrimos con el block de notas:

```
New-Item c:\build\Dockerfile -Force
```

En el siguiente paso introduciremos en el siguiente texto dentro de el archivo Dockerfile y lo guardaremos. Con esto le decimos a Docker que cree una nueva imagen con la base de microsoft/iis y ejecuta los códigos posteriores a RUN es decir, actualiza el contenido del fichero al que señalamos.

```
FROM microsoft/iis
RUN echo "Hello World - Dockerfile" > c:\inetpub\wwwroot\index.html
```

Una vez guardado y modificado el DockerFile debemos compilar la imagen, para ello usamos el comando docker build y cambiando <usuario> por nuestro ID de docker cloud.

```
docker build -t <usuario>/iis-dockerfile c:\Build
```

Para comprobar si se ha creado la imagen utilizamos el comando “docker images” y como podemos ver en la figura 9 se ha creado una imagen con mi nombre de usuario.

```
C:\Users\Administrador>docker build -t [redacted]/iis-dockerfile c:\Build
Sending build context to Docker daemon 2.048 kB
Step 1/2 : FROM microsoft/iis
 ---> b02b292ce005
Step 2/2 : RUN echo "Hello World -Dockerfile" > c:\inetpub\wwwroot\index.html
 ---> Running in 139d2c93da08
 ---> 942269057e1e
Removing intermediate container 139d2c93da08
Successfully built 942269057e1e

C:\Users\Administrador>docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
[redacted]/iis-dockerfile   latest              942269057e1e        3 minutes ago       10.5 GB
iismodificado               latest              6d070a070c8e        4 hours ago         10.7 GB
microsoft/dotnet-samples    dotnetapp-nanoserver 6313d7c84df3       18 hours ago        1.13 GB
microsoft/iis               latest              b02b292ce005        3 weeks ago         10.5 GB
```

Figura 9: Creación y comprobación de la imagen de dockerfile

Una vez creada la imagen la implementamos y ya podemos ir al navegador para comprobar que los cambios realizados sobre la imagen se han aplicado con éxito, tal y como se muestra en la figura 10:

```
docker run -d -p 80:80 <usuario>/iis-dockerfile ping -t localhost
```

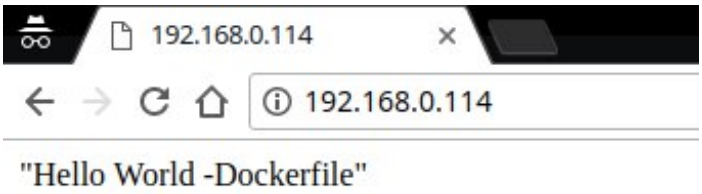


Figura 10: Contenedor modificado con dockerfile

Llegados a este punto, hemos implementado un contenedor con una imagen descargada y además hemos modificado esa imagen. Solo falta “exportar” esa imagen modificada para que pueda ser descargada por otros host de microsoft e implementarla en sus equipos. Para realizarlo usaremos Docker Hub [7].

Para cargar nuestras imágenes a Docker hub lo primero que debemos hacer es iniciar sesion en nuestra cuenta de Docker mediante el comando “docker login” donde se nos solicitara nuestro nombre de usuario y contraseña.

Una vez iniciada la sesión podemos cargar nuestra imagen en docker hub con el comando “docker push”.

```
docker push doskoy93/iis-dockerfile
```

Y con eso ya estaría disponible para descargar desde otro equipo. Para mostrar como se añadiría esa misma imagen o cualquier otra de otro usuario, vamos a borrar la imagen con el siguiente comando:

```
docker rmi <usuario>/iis-dockerfile -f
```

Ahora podemos ver que la imagen ya no aparece con el comando “docker images”, el siguiente paso es descargarla con “docker pull”:

```
docker pull <usuario>/iis-dockerfile
```

Todo esto se puede observar en la figura 11.

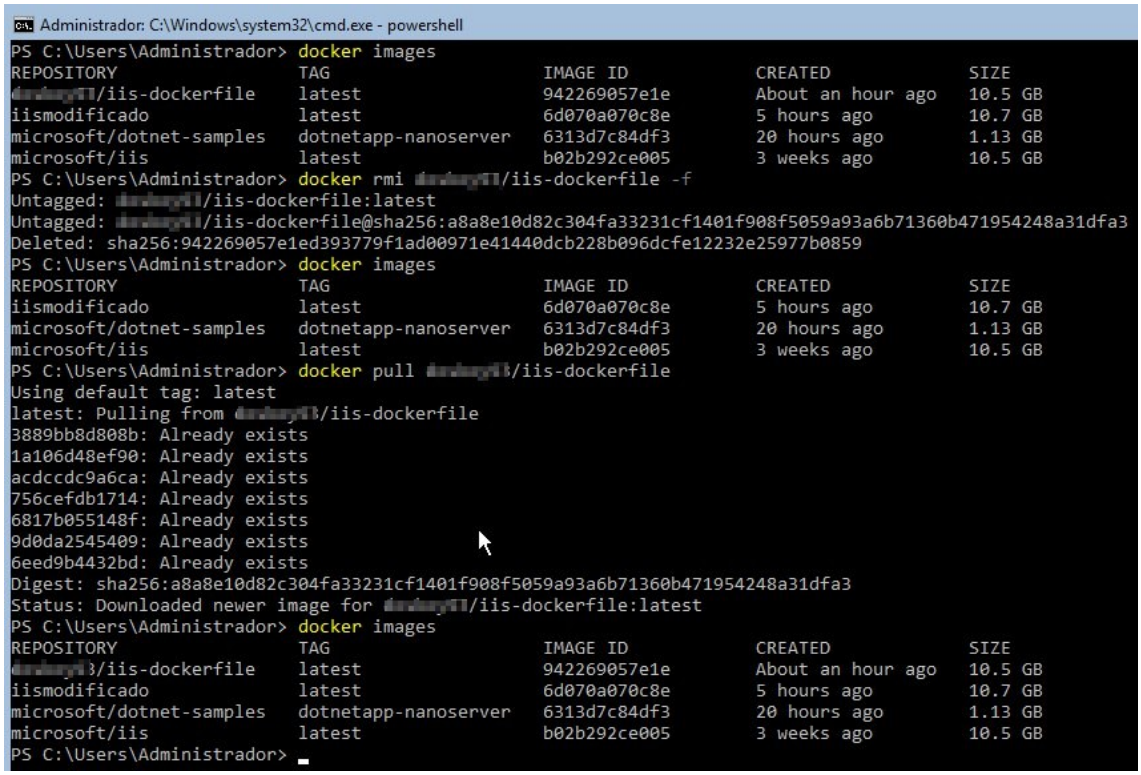


Figura 11: Descarga de imagen desde Docker hub

7. Conclusión

A través del trabajo realizado hemos podido aportar una pequeña idea sobre las posibilidades y los beneficios de trabajar con contenedores además de aportar unos pasos básicos a tomar si decidimos trabajar con ellos.

Como conclusión puedo decir que me parece una herramienta bastante útil para cualquier desarrollador ya que docker nos da la posibilidad de levantar un entorno especifico en el que probar versiones concretas de nuestra aplicación sin tener que realizar cambios en nuestro programa y además de esto, permite que cualquier otra persona desde cualquier lugar despliegue exactamente el mismo docker sin hacer ninguna configuración especial.

8. Bibliografía

[1] https://azure.microsoft.com/es-es/blog/containers-docker-windows-and-trends/

[2] https://www.theregister.co.uk/2015/08/31/hands_on_with_windows_server_2016_containers/

[3] https://docs.microsoft.com/es-es/virtualization/windowscontainers/about/

[4] https://docs.microsoft.com/es-es/virtualization/windowscontainers/quick-start/quick-start-windows-server

[5] https://hub.docker.com/r/microsoft/iis/

[6] https://docs.microsoft.com/es-es/virtualization/windowscontainers/manage-docker/manage-windows-dockerfile

[7] https://hub.docker.com/