Proyecto
ASIR 14-15

Contenedor de aplicaciones: Docker

Luis Ángel Sánchez Lasso

23/06/2015

Contenedor de aplicaciones: Docker: Docker

Contenido

Introducción 4
Objetivos 4
Definición de contenedores de Linux 5
Opciones para crear Contenedores Linux 6
Docker 6
Rocket 6
Explicación de la Herramienta Docker 7
Historia 7
Funcionamiento 7
Usos de Docker 8
Requisitos previos: 8
Hardware 8
Windows 8
Linux 8
Instalación Docker 8
Windows 8
Linux 10
Pruebas de ejecución y funcionamiento 11
Creación de contenedores 11
Comando: 11
Dockerfile 15
Alojamiento y ejecución de contenedores en la nube 17
Alojamiento de Contenedores: 17

Administrador de Nodos 19
Tutum 19
Digital Ocean 20
Microsoft Azure 20
Conclusión 21

Introducción

A continuación desarrollaré, un proyecto dedicado los contenedores de Linux, que es un método de virtualización a nivel sistema operativo, para ejecutar múltiples servidores aislados (contenedores) en un sólo Host controlador. LXC no provee una máquina virtual, pero si provee un entorno virtual que tiene sus propios espacios de proceso y red. Es similar a chroot, pero ofrece mucho más aislamiento.

Objetivos

- Definición de contenedores
- Investigar opciones disponibles de contenedores Linux.
- Explicación de la herramienta Docker.
- Desarrollo de Docker en diferentes plataformas.
- Descripción de creación y utilidad de los contenedores.
- Alojamiento y ejecución de contenedores en la nube

Definición de contenedores de Linux

Un contenedor es simplemente un proceso para el sistema operativo que, internamente, contiene la aplicación que queremos ejecutar y todas sus dependencias. La aplicación contenida solamente tiene visibilidad sobre el sistema de ficheros virtual del contenedor y utiliza indirectamente el kernel del sistema operativo principal para ejecutarse.

Podemos trazar un paralelismo entre el contenedor y una máquina virtual: ambos son sistemas auto contenidos que, en realidad, utilizan un sistema superior para ejecutar sus trabajos. La gran diferencia es que una máquina virtual necesita contener todo el sistema operativo mientras que un contenedor aprovecha el sistema operativo sobre el cual se ejecuta.

Máquinas Virtuales	Contenedores Docker
Una máquina física que aporte el hardware	Una máquina física o virtual
Un sistema operativo "Host" sobre esta máquina física	Un sistema operativo sobre esta máquina
Un sistema de virtualización o hypervisor que gestione las peticiones al hardware virtual y las ejecute sobre el real	El motor de Docker instalado en esta máquina
Un sistema operativo "Guest" bajo el hypervisor. Este sistema debe ser completo ya que no puede obtener recursos del kernel de su Host	Un contenedor basado en una imagen que contenga el motor de base de datos y todas sus dependencias
Instalar bajo el sistema "Guest" el motor de base de datos y todas sus dependencias	

Opciones para crear Contenedores Linux

Docker

Docker es una plataforma abierta para los desarrolladores y administradores de sistemas para la construcción, envío y ejecución de aplicaciones. Consta de Motor de Docker, una Herramienta ligera y portátil y Docker Hub que es un servicio en la nube para compartir aplicaciones que permite la automatización de flujos de trabajo. Docker permite ensamblar aplicaciones rápidamente y elimina la fricción entre los entornos de desarrollo, control de calidad y producción. Como resultado, se puede enviar más rápido y ejecutar la misma aplicación, sin cambios, en los ordenadores portátiles, los centros de datos de máquinas virtuales, y cualquier nube.

La característica principal del sistema es que esos módulos se pueden agregar o quitar dependiendo de tus necesidades.

Rocket

Es otra posibilidad o elección para la ejecución Docker, también está diseñado para entornos de servidores con los requisitos de seguridad, enlazamiento, velocidad y de producción más resueltos.

El software se compone de dos elementos, cada uno de los cuales es una herramienta de línea de comandos sencilla y autónoma.

1). Actool: Es el primer componente que administra la construcción de contenedores. Incluso trata la validación de contenedor y descubrimiento.

2). Rkt: Se llama así, como todos los comandos UNIX principales utilizados en Rocket son tres letras. Le ayuda en el cuidado de la recuperación por y ejecutar imágenes de contenedores.

Al contrario del enfoque de Docker, Rocket no implica proceso exterior y estas herramientas no son sólo una interfaz de usuario para conversar a cualquier otro servidor.
Siempre que llames a rkt para ejecutar un contenedor lo hará de forma rápida dentro de la gama de su propio árbol de procesos y cgroup.

Explicación de la Herramienta Docker

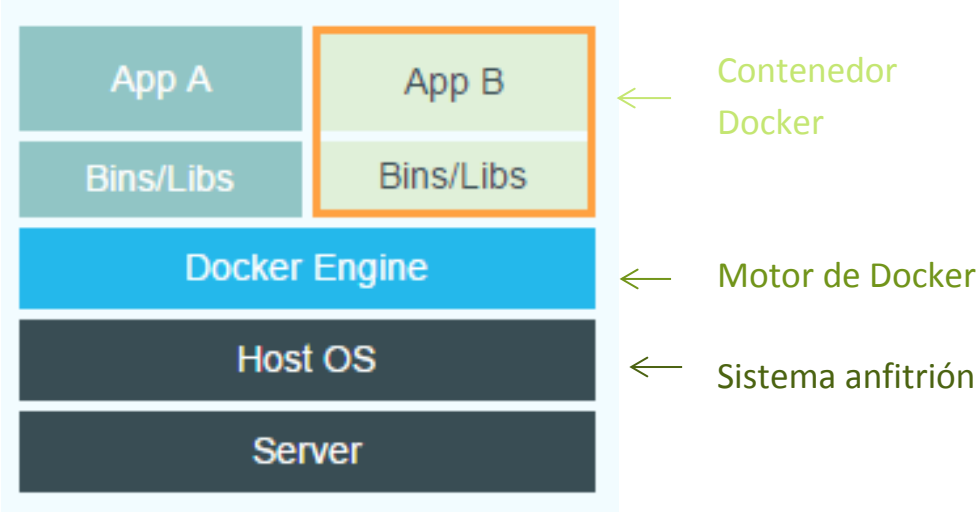
Historia

Salomón Hykes, comenzó Docker como un proyecto interno dentro dotCloud, una plataforma como un servicio de la empresa, con las contribuciones iniciales por otros ingenieros dotCloud incluyendo Andrea Luzzardi y Francois-Xavier Bourlet. Jeff Lindsay también participó como colaborador independiente. Docker representa una evolución de la tecnología patentada de dotCloud, que a su vez construyen en proyectos de código abierto anteriores como Cloudlets.

Funcionamiento

Con la tecnología de Docker podremos virtualizar un Linux con todas las aplicaciones que necesitemos dentro de nuestro sistema operativo Linux, para **"empaquetarlo y desplegarlo"** en cualquier otro Linux sin necesidad más que de introducir un par de comandos.

Docker implementa un alto nivel API para proporcionar contenedores ligeros que se ejecutan los procesos de manera aislada. Basándose en la cima de las instalaciones que ofrece el kernel Linux un contenedor Docker, en contraposición a una máquina virtual tradicional, lo hace no exigen o incluyen un sistema operativo independiente. En su lugar, se basa en la funcionalidad del núcleo y utiliza aislamiento de recursos (CPU, memoria, bloque de E / S, red, etc.) y los espacios de nombres separados para aislar completamente la vista de la aplicación de la sistema operativo. Docker accede a la virtualización del kernel Linux ofrece, ya sea directamente a través de la proporcionada biblioteca “libcontainer”.



Usos de Docker

Ya que los contenedores te dan un ambiente aislado del resto del sistema, las posibilidades de trabajo incluyen:

- Empaquetamiento y despliegue de aplicaciones automatizado y controlado.
- Creación Ambientes de PaaS.
- Testing e integración continua
- Despliegue y escalamiento de aplicaciones y bases de datos.

Requisitos previos:

Hardware

El procesador debe soportar la virtualización por hardware.

Windows

Docker ha sido probado en Windows 7.1 y 8, si se quiere ejecutar en versiones anteriores, el procesador debe soportar la virtualización por hardware.

Linux

Las distribuciones que funcionan con versiones iguales o superiores al kernel 3.8 de Linux.

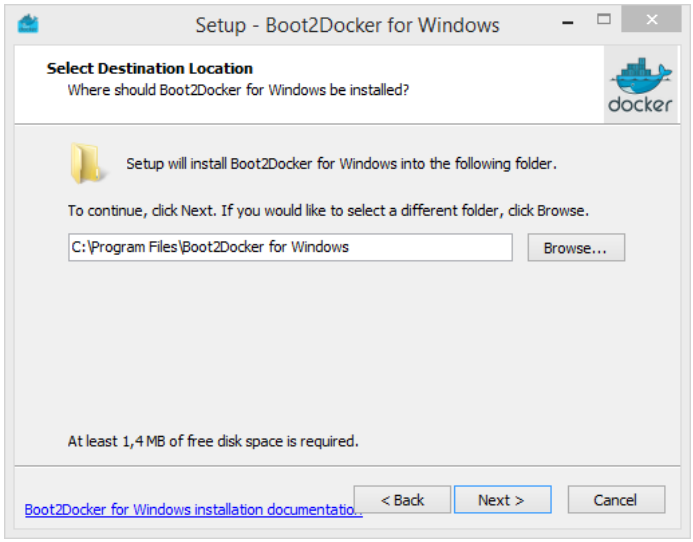
Instalación Docker

Windows

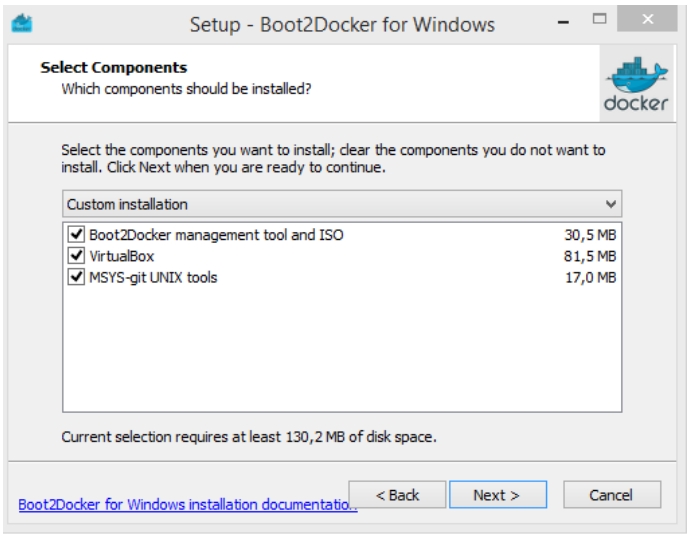
Para la instalación descargaremos el fichero de ejecución
https://github.com/boot2docker/windows-installer/releases/tag/v1.5.0

Instalación

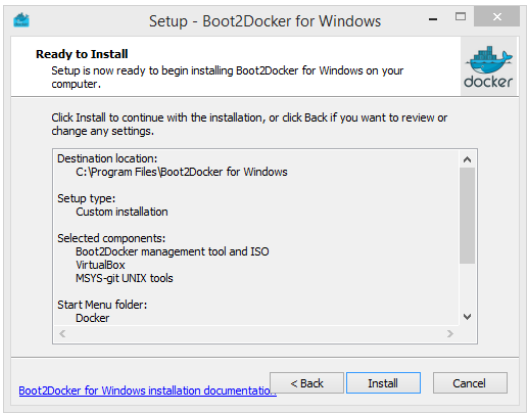
Simplemente ejecutamos el archivo que hemos descargado de la página web, nos abrirá una ventana con la instalación de Docker



Nos preguntará el lugar donde instalaremos Docker



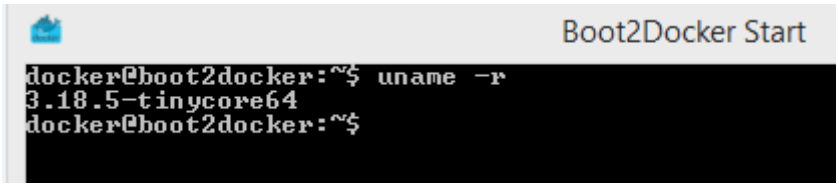
Lo siguiente que preguntará el programa será qué queremos instalar, lo que hace Docker en Windows es, crear una máquina virtual a la cual le instala un sistema operativo Linux "Boot2Docker", ya que Docker necesita utilizar el kernel de Linux, para esto utiliza el programa de virtualización VirtualBox, también instala la herramienta msys-git que utiliza como terminal para conectarse a la máquina virtual que creará.



Luego para su ejecución creará un acceso directo en el escritorio

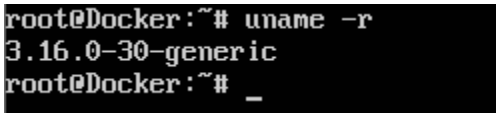
Al ejecutar la aplicación, lo que hace es verificar si existe la máquina virtual en Docker si no existe, la crea, y si existe, se inicia la máquina virtual y se conecta mediante la creación de una clave de conexión con ssh.

El sistema que corre es un sistema Linux con el siguiente kernel:



Linux

Ahora vamos con la Instalación en Linux, para la cual utilizaré la distribución Ubuntu Server 14.04 con el kernel 3.16



Para su instalación simplemente ejecutamos el comando siguiente

```
sudo aptitude install docker.io
Se instalarán los siguiente paquetes NUEVOS:
  aufs-tools{a} cgroup-lite{a} docker.io git{a} git-man{a} liberror-perl{a}
0 paquetes actualizados, 6 nuevos instalados, 0 para eliminar y 43 sin actualizar.
Necesito descargar 7.553 kB de archivos. Después de desempaquetar se usarán 46,6 MB.
¿Quiere continuar? [Y/n/?] Y_
```

El programa no requiere de una configuración, simplemente con la instalación es suficiente.

Pruebas de ejecución y funcionamiento

En esta parte necesitamos entender algunos conceptos previos

Contenedor: Son como directorios, contienen todo lo necesario para que una aplicación pueda funcionar sin necesidad de acceder a un repositorio externo al contenedor. Cada uno de éstos es una plataforma de aplicaciones segura y aislada del resto que podamos encontrar o desplegar en la misma máquina host.

Imágenes: La imagen Docker podríamos entenderla como un SO con aplicaciones instaladas (Por ejemplo un OpenSUSE con un paquete ofimático). Sobre esta base podremos empezar a añadir aplicaciones que vayamos a necesitar en otro equipo donde tengamos intención de usar la imagen.

Repositorios: También conocidos como Registros Docker, contienen imágenes creadas por los usuarios y puestas a disposición del público. Podemos encontrar repositorios públicos y totalmente gratuitos o repositorios privados donde tendremos que comprar las imágenes que necesitemos. Estos registros permiten desarrollar o desplegar aplicaciones de forma simple y rápida en base a plantillas, reduciendo el tiempo de creación o implementación de aplicaciones o sistemas.

Creación de contenedores

Comando:

Para la creación de un contenedor mediante comando debemos de seguir los siguientes pasos:

Formato de comando: docker [acción] [opciones] [Imagen/Contenedor]

1- **Obtener imagen:**
   Primero debemos saber sobre que distribución queremos montar nuestra aplicación, por ejemplo usaremos debian, sabiendo esto podemos buscar las diferentes imágenes utilizando el comando siguiente:

```
usuario@docker:~$ docker search debian
NAME                                    DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
debian                                  (Semi) Official Debian base image.              390       [OK]
google/debian                                                                           40                   [OK]
neurodebian                             NeuroDebian provides neuroscience research...   6         [OK]
tutum/debian                            Debian image with SSH access. For the root...   5                    [OK]
hanswesterbeek/google-debian-oracle-jdk Oracle's JDK installed on top of Google's ...   5                    [OK]
azul/zulu-openjdk-debian                Zulu is a fully tested, compatibility veri...   4                    [OK]
mschuerig/debian-subsonic               Subsonic 5.1 on Debian/wheezy.                  3                    [OK]
shuron/debian-openjdk-7                 Open JDK 7 64x on plain debian (Jessie) La...   2                    [OK]
webhippie/cedarish-debian               Heroku cedar-ish base images for Docker bu...   1                    [OK]
```

Para obtener una imagen (descargaremos la imagen debian 7) ejecutaremos el siguiente comando:

```
usuario@docker:~$ docker pull debian:7
Pulling repository debian
c9fa20ecce88: Pulling dependent layers
511136ea3c5a: Pulling fs layer
```

Pull: acción para descargar desde el repositorio.
debian: nombre de la imagen a descargar.
7: versión de la imagen.

2- Ver las imágenes creadas:

```
usuario@docker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
debian              7                   c9fa20ecce88        6 days ago          84.98 MB
```

Vemos la imagen que hemos descargado, la versión, y un ID que le asigna Docker, hace cuanto se creó y el tamaño que ocupa.

3- Usando la imagen

Ahora lo que procedemos a hacer es usar la imagen que tenemos, para ello debemos ejecutar el siguiente comando:

```
usuario@docker:~$ docker run -t -i -p 80:80 debian:7
root@b3fe6abd844b:/#
```

run: ejecutar un comando en una imagen
-t: abrir un terminal
-i: activa el modo interactivo.
-p: opción para seleccionar el puerto que publicara el contenedor.
80:80: puerto del sistema operativo principal: puerto del contenedor
Debian: 7: imagen a ejecutar
/bin/bash: comando que se ejecutará en la imagen

4- Instalaciones necesarias:

Ahora podemos utilizar el nuevo terminal del contenedor para la instalación de paquetes, de momento instalaré apache para comprobar que funciona como esperamos. Pero antes actualizamos la lista de paquetes

```
root@b3fe6abd844b:/# apt-get update
Get:1 http://security.debian.org wheezy/updates Release.gpg [836 B]
Get:2 http://security.debian.org wheezy/updates Release [102 kB]
Get:3 http://http.debian.net wheezy Release.gpg [1655 B]
Get:4 http://http.debian.net wheezy-updates Release.gpg [836 B]
Get:5 http://http.debian.net wheezy Release [168 kB]
Get:6 http://security.debian.org wheezy/updates/main amd64 Packages [369 kB]
```

Y luego instalamos el paquete apache2

```
root@b3fe6abd844b:/# apt-get install -y apache2
```

Ahora iniciamos el servicio

```
root@b3fe6abd844b:/# service apache2 start
```

Y en el fichero .bashrc en /root/.bashrc añadiremos la línea siguiente

```
/usr/sbin/apache2ctl start
```

Con esto conseguimos que se inicie el servicio de apache cada vez que ejecutemos el contenedor

Lo que haremos será presionar el conjunto de teclas Crtl + P y luego Ctrl + Q para no matar el proceso y luego salir.

Ahora si ejecutamos el comando para ver los contenedores veremos que se está ejecutando el contenedor que hemos creado previamente:

```
usuario@docker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                   NAMES
b3fe6abd844b        debian:7            /bin/bash           44 minutes ago      Up 44 minutes       0.0.0.0:80->80/tcp      loving_wozniak
```

ps: sirve para listar los contenedores, si lo usamos sin opciones aparecerán los contenedores que están ejecutándose lo vemos en la columna de STATUS. Para ver todos los contenedores, tanto los que se están ejecutando como los que no, utilizamos la opción -a.

5- Guardando contenedores

Ahora ya tenemos el contenedor, lo que haremos será guardar el estado y convertirlo a una imagen, con el siguiente comando:

```
suario@docker:~$ docker commit b3fe6abd844b proasir/apache2
18ec92630c9c020a54038e149fd8f2ed2b61fa84bc225ffe27a998fb3c51b65
```

commit: acción que usamos para guardar estado contenedores y convertirlos en una imagen

b3fe6abd844b: es el ID del contenedor que se docker selecciona aleatoriamente

proasir/apache2: es el “nombre” de la imagen, el formato que se utiliza es usuario-docker/nombre-de-imagen

Docker generará un código aleatorio que será el ID de la imagen que es el que vemos a continuación del comando

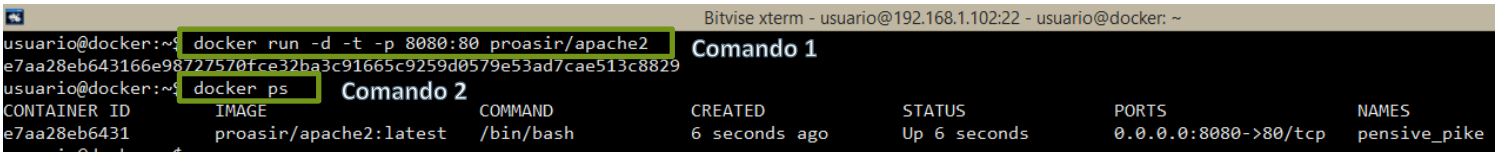
Luego podemos ver la imagen que hemos creado ejecutando el siguiente comando:

```
usuario@docker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
proasir/apache2     latest              418ec92630c9        29 minutes ago
debian              7                   c9fa20ecce88        8 days ago
```

Vemos que tenemos la imagen debian y la que hemos creado para realizar una prueba.

6- Ejecutando la nueva imagen:

Ahora tenemos nuestra imagen lista para la ejecución, lo que haremos será ejecutar un comando parecido al anterior pero esta vez le indicaremos el puerto que usaremos para publicar la aplicación:



Comando 1:

run: acción para correr una imagen

-t: iniciar nuevo terminal

-i: terminal modo interactivo (esto no es necesario)

-d: para ejecutar el contenedor en segundo plano

-p: opción para seleccionar el puerto que publicara el contenedor.

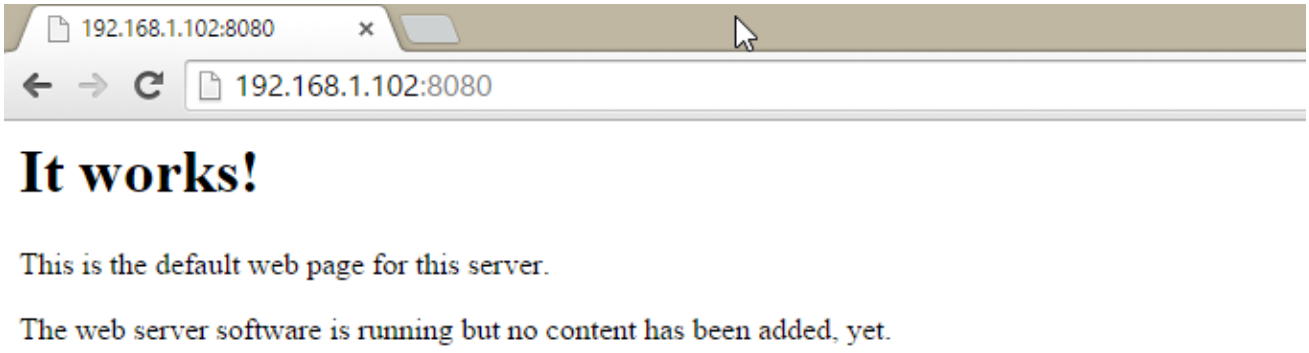
8080:80: puerto del sistema operativo principal: puerto del contenedor

proasir/apache2: nombre de la imagen

Comando 2:

ps: listar los contenedores que se están ejecutando.

Ahora si accedemos desde un navegador a la IP del sistema principal, veremos que está ejecutando el contenedor

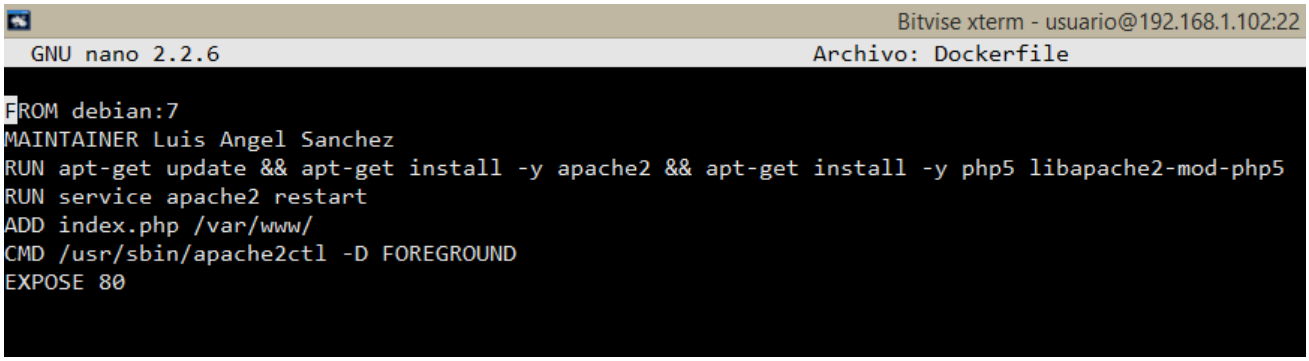


La IP que utilizamos para acceder al contenedor es la del sistema principal ya que docker crea reglas de enrutamiento con iptables.

Dockerfile

Ahora haremos lo mismo pero creando un Dockerfile, que es un script que utilizaremos para automatizar la creación del contenedor, luego de tenerlo, lo guardaremos. Esta vez tendremos instalado apache y php5

Dockerfile completo:



FROM: Aquí colocaremos la imagen base con la que queremos trabajar en este caso es un debian 7

MAINTAINER: Aquí el autor del nuevo contenedor.

RUN: Con esta orden le diremos los comandos que se ejecutará dentro, normalmente se ejcuta para instalación de programas.

ADD: Esta orden podremos añadir algún archivo o directorio, en este caso lo que añadiremos es un fichero para comprobar que está funcionando apache junto con PHP.

CMD: Este podemos ejecutar un comando igual a RUN pero este es un comando que se ejecutará al iniciar el contenedor.

EXPOSE: Con este seleccionamos el puerto que se publicara

Ahora lo que nos toca es crear el contenedor, con el comando siguiente:

```
usuario@docker:~$ docker build -t proasir/apache-php .
Sending build context to Docker daemon 36.35 kB
Sending build context to Docker daemon
Step 0 : FROM debian:7
 ---> c9fa20ecce88
Step 1 : MAINTAINER Luis Angel Sanchez
 ---> Running in 890f0f68efc3
 ---> 5c2803dfbfbc
Removing intermediate container 890f0f68efc3
Step 2 : RUN apt-get update && apt-get install -y apache2 && apt-get install -y php5 libapache2-mod-php5
 ---> Running in 9e4654223b36
Get:1 http://security.debian.org wheezy/updates Release.gpg [836 B]
Get:2 http://security.debian.org wheezy/updates Release [102 kB]
```

Podemos ver que se ha creado la imagen

```
usuario@docker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
proasir/apache-php  latest              b7b9a00e305b        40 minutes ago      170.2 MB
proasir/apache2     latest              af84ba453ee3        44 hours ago        147.1 MB
debian              7                   c9fa20ecce88        11 days ago         84.98 MB
usuario@docker:~$
```

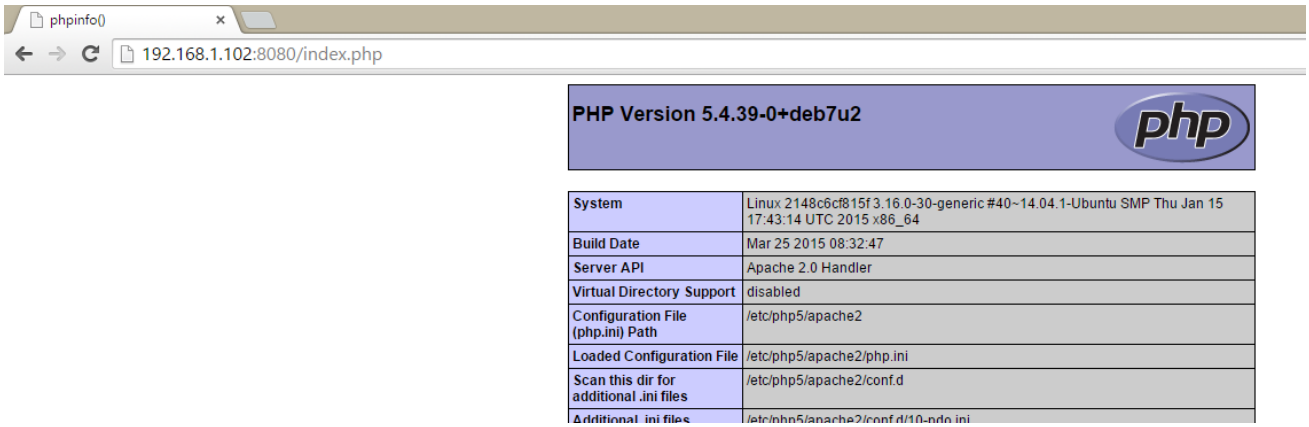
Ahora podemos utilizar esta imagen para ello ejecutamos el comando de la siguiente forma:

```
usuario@docker:~$ docker run -t -i -d -p 8080:80 proasir/apache-php
2148c6cf815f770776068dd3a49c3362ee4dbe55a64c41ff50af023098340cd4
```

Vemos los contenedores que están ejecutándose:

```
usuario@docker:~$ docker ps
CONTAINER ID        IMAGE                      COMMAND                CREATED             STATUS              PORTS                    NAMES
2148c6cf815f        proasir/apache-php:latest  /bin/sh -c '/usr/sbi   2 minutes ago       Up 2 minutes        0.0.0.0:8080->80/tcp     focused_mayer
```

Y con esto podemos acceder a lo que ofrece el contenedor mediante el puerto 8080 de nuestro equipo:



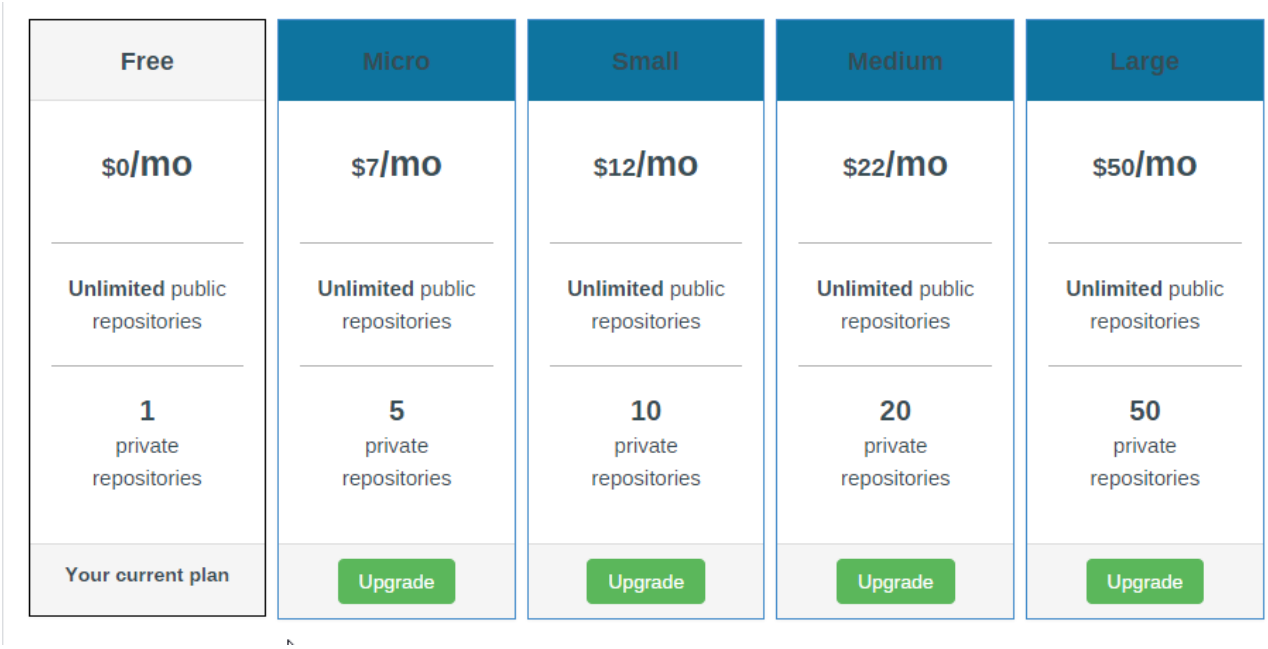
Alojamiento y ejecución de contenedores en la nube

Para el alojamiento y ejecución de contenedores en la nube tenemos algunas empresas que nos no ofrecen, como la que detallo a continuación:

Alojamiento de Contenedores:

Para el alojamiento de contenedores tenemos la página https://hub.docker.com que nos permite guardar los contenedores de forma gratuita.

Estos contenedores que almacenamos son públicos y cualquier usuario puede descargarlos y utilizarlo al igual que podemos hacer los contenedores de otros usuarios, si queremos podemos tener nuestros contenedores almacenados de forma privado con los planes que nos ofrece Docker



Para alojar o descargar las imágenes o contenedores que ya tenemos creados, debemos estar registrados en la página de docker luego podemos acceder a docker de la siguiente forma:

```
usuario@docker:~$ docker login
Username: proasir15
Password:
Email: proasir15@yopmail.com
Login Succeeded
```

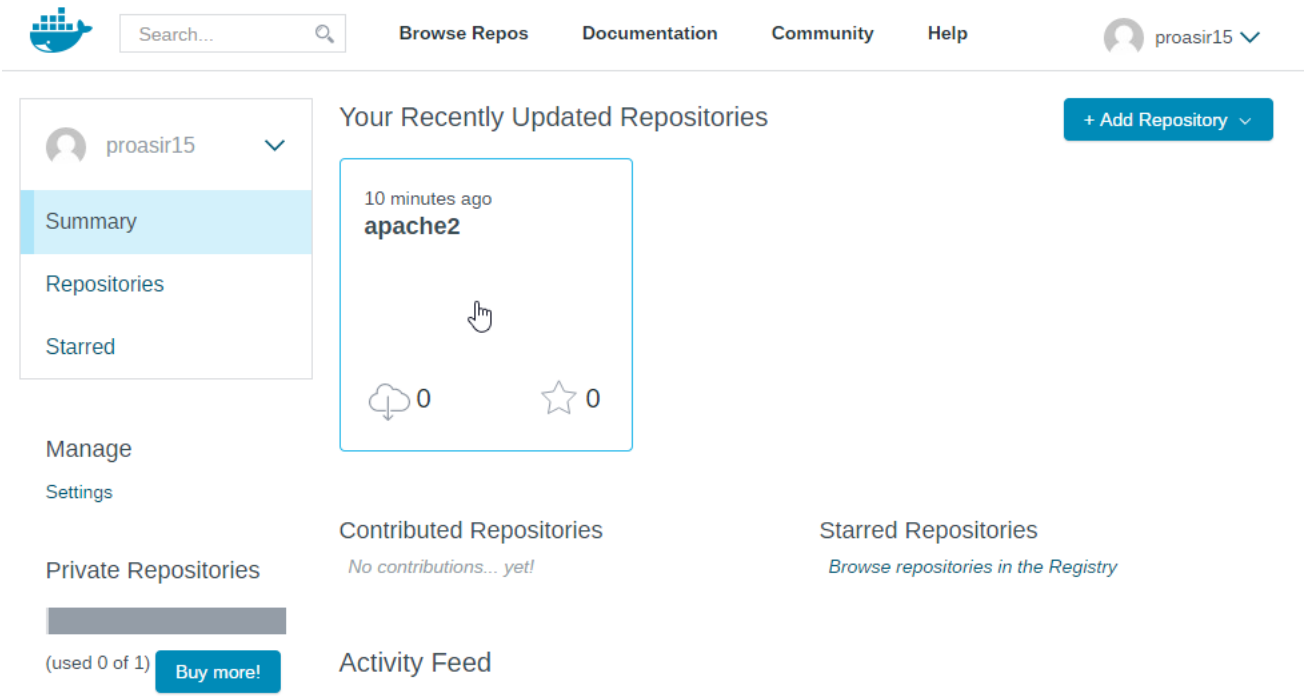
Así podemos subir las imágenes y tener nuestros contenedores disponibles cuando queramos.
Ahora para subir una imagen simplemente ejecutamos el siguiente comando:

```
usuario@docker:~$ docker push proasir15/apache2
The push refers to a repository [proasir15/apache2] (len: 1)
Sending image list
Pushing repository proasir15/apache2 (1 tags)
511136ea3c5a: Image already pushed, skipping
e977d53b9210: Image already pushed, skipping
c9fa20ecce88: Image already pushed, skipping
5c2803dfbfbc: Image successfully pushed
0f1a322b3032: Buffering to disk 18.18 MB
```

De esta forma empezará a almacenar la imagen.

Algo IMPORTANTE es que la parte en la que colocamos el nombre de la imagen NO PUEDE SER DE OTRO USUARIO, debe ser el usuario con el cual nos hemos accedido al hub de Docker, si lo hacemos de otra forma que no sea nuestro usuario no nos permitirá almacenar esta imagen, si nos encontramos en este caso podemos ejecutar la imagen con el comando “run” y luego guardar el estado de la misma con “commit” con el formato usuario/nombre-imagen[:versión].

Si volvemos a la página, veremos que tenemos el contenedor disponible

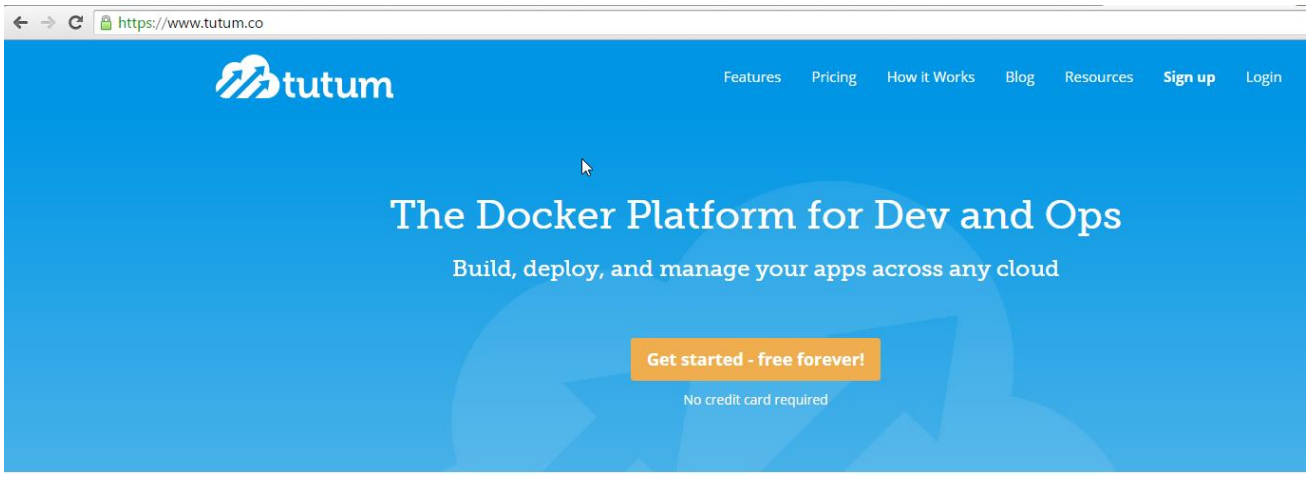


Ahora este lo podemos descargar en otro equipo que tenga Docker instalado y utilizarlo.

Administrador de Nodos

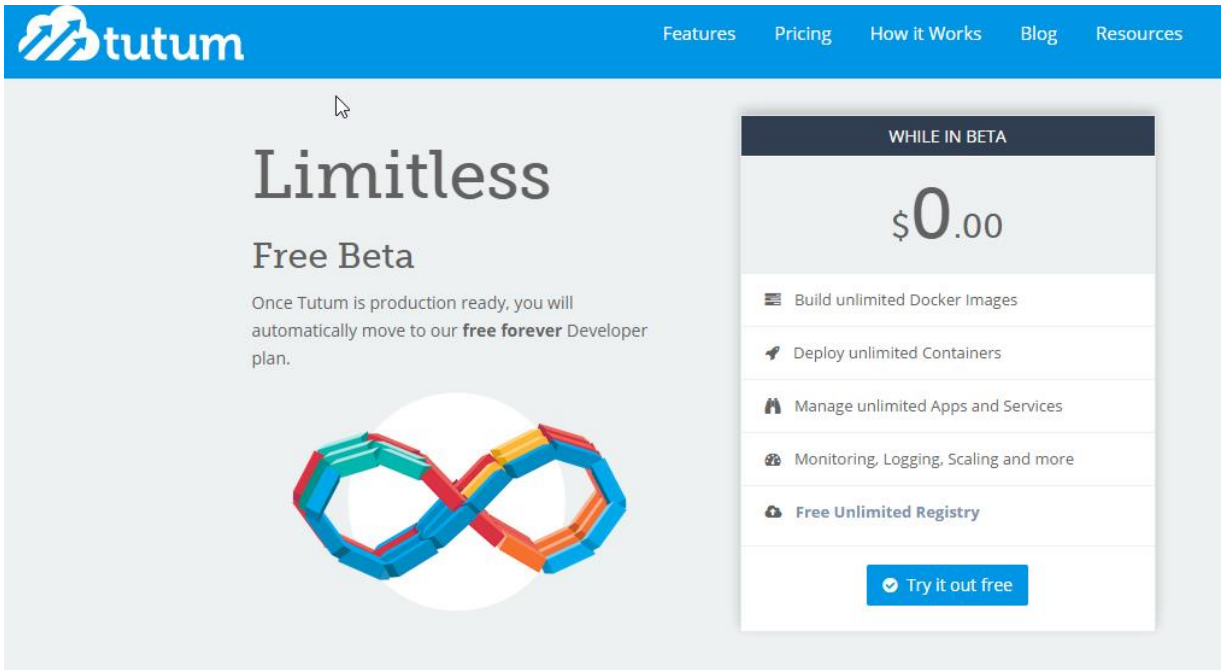
Ahora tenemos lo que tenemos una empresa que nos ofrece la administración de nuestros nodos

Tutum



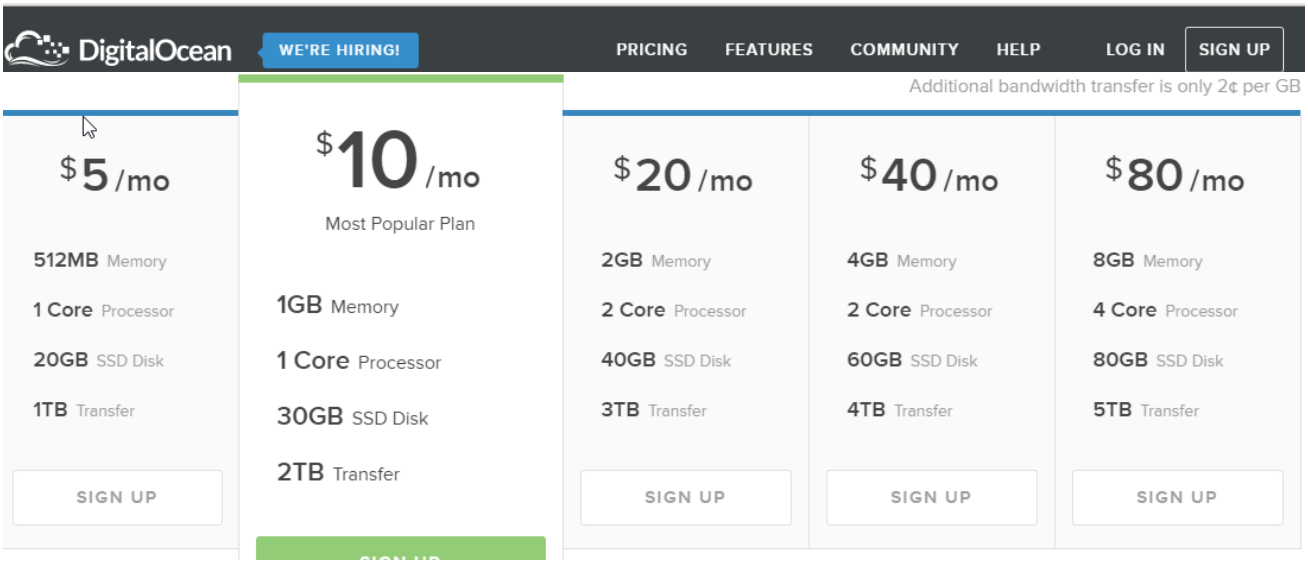
Tutum hace que sea fácil de crear y ejecutar el código. Proporcionan a todos los usuarios con un registro gratuito, almacenar sus aplicaciones en contenedores. Acceso rápido y la selección de Imágenes optimizadas de Tutum para comenzar a ejecutar.

Momentáneamente tutum es gratuito, ya que está en periodo de desarrollo (Beta).

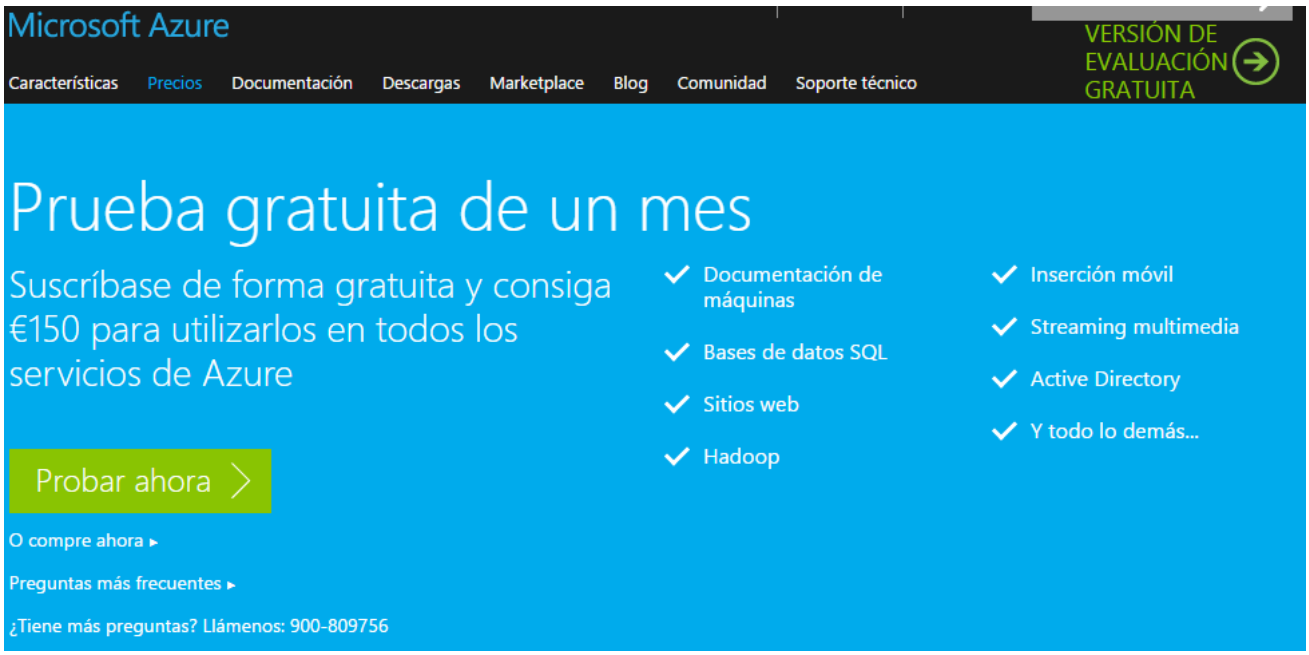


Desde Tutum podemos administrar los nodos de servicios que ya hayamos contratado con las siguientes empresas:

Digital Ocean



Microsoft Azure



Conclusión

En este proyecto he podido aprender una forma fácil de poder dar servicio a los programadores, con una herramienta tan útil como lo es Docker.

También en la búsqueda de información me cruce con ofertas de trabajo que se necesitara experiencia con Docker o temas de Contenedores Linux.

Este es un tema algo sencillo pero de mucha utilidad para nosotros como administradores informáticos. Recomiendo este tema se de en cursos futuros ya es muy útil para los administradores de sistemas, en aligerar y simplificar procesos de virtualización.

Muchas Gracias.

Luis Ángel Sánchez Lasso.