08_introduccion_a_docker

October 15, 2020

0.1 Antecedentes.

Docker es una herramienta que aprovechaba inicialmente las ventajas de GNU/Linux y bibliotecas tales como `cgroups` para crear y gestionar contenedores.

En versiones previas, Docker estaba basado en LXC, pero a partir de la versión 0.9, Docker utiliza su propia biblioteca llamada `libcontainer`, la cual es independiente del kernel de GNU/Linux y puede ser ejecutada desde Windows con Hyper-V.

En el caso de ser utilizado en un entorno GNU/Linux, Docker es instalado como un servicio del sistema.

Recientemente, Docker ha aprovechado las capacidades Hyper-V de Windows para ofrecer contenedores nativos. Sin embargo, Hyper-V no está disponible en versiones básicas de Windows como la versión Home.

Para utilizar Docker en su versión completa se requiere Windows 10 Pro con Hiper-V activado.

0.2 Ediciones de Docker.

Docker ofrece una versión empresarial de su herramienta llamada *Docker Enterprise Edition (EE)* y una versión de libre distribución llamada *Docker Desktop*, también conocida como *Docker Community Edition (CE)*.

0.2.1 *Docker Desktop for Windows y Mac.*

En el sitio de Docker Desktop, están disponibles los paquetes de instalación para sistemas Windows superiores a la versión Windows 10 Pro, el cual aprovecha las funcionalidades de Hyper-V, asi como para OS X.

Advertencia: para poder ejecutar *Docker Desktop* en Windows 10 Pro, es necesario habilitar Hyper-V (el hipervisor nativo de Windows), el cual puede provocar que algunos otros hipervisores tales como Virtualbox sean inutilizables.

Los detalles de requerimientos para *Docker Desktop* pueden ser consultados en https://docs.docker.com/docker-for-windows/install/.

0.2.2 *Docker Toolbox.*

En el caso de no contar con los requisitos mínimos para instalar *Docker Desktop* para Windows, está disponible Docker Toolbox, el cual utiliza Virtualbox en vez de Hyper-V.

Los detalles de instalación están disponibles en https://docs.docker.com/toolbox/overview/

0.3 Instalación de docker en GNU/Linux.

0.3.1 Instalación de Docker CE en Ubuntu.

En el caso de Ubuntu, sólo es necesario ejecutar:

`apt install docker.io`

[ ]:
```
sudo apt update && sudo apt upgrade -y && sudo apt install docker.io -y
```

0.3.2 Instalación de *Docker CE* en CentOS 7.

El sitio de descargas de *Docker CE* ofrece paquetes de instalación para diversas plataformas.

En el caso de CentOS 7 es posible utilizar el repositorio de Docker para instalar la herramienta.

- Es necesario instalar algunas dependencias previas.

[ ]:
```
sudo yum install yum-utils device-mapper-persistent-data lvm2 -y
```

- Se debe de dar de alta al repositorio.

[ ]:
```
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/
↪docker-ce.repo
```

[ ]:
```
sudo yum update -y
```

- Se instalará el software.

[ ]:
```
sudo yum install docker-ce -y
```

0.4 Inicio del servicio de Docker.

- Se iniciará y habilitará el servicio.

[ ]:
```
sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl status docker --no-pager
```

[ ]:
```
docker -v
```

0.5 *Docker Hub.*

Docker basa su funcionalidad en el uso de "imágenes" de contenedores creadas previamente, las cuales son accedidas desde un repositorio. Dicho repositorio es conocido como *Docker Hub* y está localizado en https://hub.docker.com.

Es posible explorar la biblioteca de imágenes desde https://hub.docker.com/explore.

0.6 Ejecución de comandos de Docker.

El comando `docker` permite acceder a funciones de ejecución y gestión de contendores desde una terminal con uin sintaxis como:

```
docker <subcomando> <opciones y argumentos>
```

0.7 Ejecución de un contenedor.

Para acceder y ejecutar un contenedor sólo es necesario utilizar el siguiente comando con el nombre de la imagen que se desea ejecutar.

```
docker run  <argumentos> <etiqueta> <comando>
```

- Docker buscará la imagen, la descargará y la ejecutará.
- Tan pronto como el contenedor termine la ejecución, éste será terminado, pero no destruido.

Los argumentos de ejecución de un contenedor permiten definir muchas características de éste, tales como el uso de recursos, conexiones y la interacción con el usuario.

Referencia: https://docs.docker.com/engine/reference/commandline/run/

NOTA: Es necesario que el usuario tenga los permisos suficientes para ejecutar el comando.

Ejemplo:

Se ejecutará la imagen https://hub.docker.com/_/hello-world/

[ ]:
```
sudo docker run hello-world
```

0.7.1 Acceso a un contenedor en ejecución mediante la CLI.

Para poder acceder a un contenedor mediante un emulador de terminal se utilizan los parámetros:

- `-i`, el cual permite al usuario interactuar con un contenedor en ejecución.
- `-t`, el cual permite conectarse a un conetendor mediante una terminal.

La sintaxis es la siguiente:

```
docker run -it <nombre> <entorno de shell>
```

El entorno de shell más comunmente utilizado es `/bin/bash`.

El acceso al contenedor siempre será como `root`.

Ejemplo:

Ejecute el siguiente comando desde una terminal para acceder a un contenedor de *CentOS*.

```
sudo docker run -it centos /bin/bash
```

Para salir del contenedor teclee `exit`.

0.7.2 Definición de recursos.

Es posible dimensionar los recursos a los que puede acceder un contenedor mediante ciertos parámetros al utilizar `docker run`.

La lista commpleta de recursos puede ser consultada en https://docs.docker.com/config/containers/resource_constraints/

Ejemplo:

Los recursos más comunes que se definen para un contenedor son:

- Memoria.
- CPU.

Al ejecutar el siguiente comando eu nuna terminal, se creará un contenedor basado en Ubuntu al que se le limita com el 50% del rendimiento de un núcleo de la CPU y 128MB de RAM.

```
sudo docker run -it --cpus=".5" --memory="128m" ubuntu /bin/bash
```

0.7.3 Reenvío de puertos.

Es posible reenviar los paquetes desde un puerto del contenedor a un puerto del sistema anfitrión usando la siguiente sintaxis:

```
docker run --port <puerto del anfitrión>:<puerto del contenedor> ...
```

```
docker run -p <puerto del anfitrión>:<puerto del contenedor> ...
```

0.7.4 Volúmenes.

Es posible asignar uno o varios volúmenes a un contenedor, los cuales serán montados de forma similar a como se hace con fstab.

```
docker run -v <directorio del anfitrión>:<directorio del contenedor> ...
```

0.7.5 Ejecución de un contendor en trasfondo.

El parámetro `-d` o `--detach` ejecutará un contenedor desligado a una terminal.

0.7.6 Asignación de un nombre a un contenedor.

Para asignarle un nombre a un contenedor se su uitliza el parámetro `--name` seguido del nombre que se le desea asignar.

```
docker run --name <nombre del contendor> <argumentos> <nombre de la imagen>
<comando>
```

0.8 Recuperación de un contenedor ejecutándose en trasfondo.

Para retomar a un contendor que se está ejecutando en trasfondo se utiliza el comando:

```
docker attach <identificador>
```

Referencia: https://docs.docker.com/engine/reference/commandline/attach/

Ejemplo:

La siguiente celda ejecutará un contenedor de Ubuntu utilizando los parámetros `-dit` y se le asignará el nombre `Prueba`.

[ ]:
```
sudo docker run -idt --name Prueba centos /bin/bash
```

[ ]:
```
sudo docker ps
```

Para acceder a este contenedor utilice el siguiente comando desde una terminal:

`sudo docker attach Prueba`

Para salir del contenedor use `exit`.

0.9 Consulta sobre los contenedores en un sistema.

Es posible hacer consultas sobre el estado de los contenedores del sistema local mediante:

`docker ps`

Referencia: https://docs.docker.com/engine/reference/commandline/ps/

Al usar `docker ps` sin argumentos, el comando nos regresará el listado de contenedores en ejecución.

Los nombres de los contenedores son asignados automáticamente por Docker.

Ejemplo:

La siguiente celda desplegará el listado de contenedores en ejecución. Si no ha cerrado el contenedor del ejemplo previo, este será enlistado.

[ ]:
```
sudo docker ps
```

Al usar `docker ps -a` el comando nos regresará el listado de contenedores que hayan sido ejecutados.

Ejemplo:

La siguiente celda desplegará el listado de todos los conrtenedores que se hayan ejecutado.

[ ]:
```
sudo docker ps -a
```

Detención de un contenedor en ejecución.

Para detener un contenedor utilice el comando:

`docker stop <identificador>`

Referencia: https://docs.docker.com/engine/reference/commandline/stop/

Ejemplo:

Se detendrá el contenedor llamado `Prueba` el cual fue ejecutado previamante.

[ ]:
```
sudo docker stop Prueba
```

0.10 Eliminación de un contenedor.

Para eliminar uno o varios contenedor existentes se puede utilizar el comando:

`docker rm <identificadores>` En donde el identificador puede ser el número de ID o el nombre del contenedor.

Referencia: https://docs.docker.com/engine/reference/commandline/rm/

Ejemplo:

Se eliminará el contenedor llamado `Prueba`.

[ ]:
```
sudo docker ps -a
```

[ ]:
```
sudo docker rm Prueba
```

[ ]:
```
sudo docker ps -a
```

0.11 Despliegue de la información de docker.

Para desplegar la información del servicio de docker que corre en el sistema local se utiliza el comando:

`docker info`

Ejemplo:

[ ]:
```
sudo docker info
```