Despliegue de Oracle HTTP Server 12c sobre Containers Docker

Por Francisco Riccio

Introducción

Este artículo está enfocado a explicar el concepto básico de Containers y su implementación mediante Docker para desplegar un Oracle HTTP Server 12c (OHS).

Los containers son softwares de "virtualización" que utilizan ciertas capacidades de cómputo dentro de un sistema operativo de manera aislada para ejecutar una aplicación determinada sin importar algunos aspectos del anfitrión que lo alberga.

Algo interesante, es que los containers son ligeros y portables, porque puedo trasladarlos entre diferentes sistemas operativos con la finalidad de ejecutar nuestra aplicación sin importar los softwares que tenga instalado el Sistema Operativo anfitrión, utilizando los recursos más básicos.

Hoy los contenedores son una de las herramientas más utilizadas en un Cloud PaaS y disponible en muchos proveedores de Nube como: IBM, Oracle, Microsoft, Amazon, etc.

Entre los líderes de Containers se encuentra Docker (https://www.docker.com/), siendo una tecnología open-source y nos permite un fácil entorno de despliegue de nuestras aplicaciones mediante contenedores.

Mi implementación estará enfocada en un ambiente On-Premise y se detallará todas las fuentes publicadas por Oracle para el correcto soporte y certificación de su producto sobre esta tecnología.

Implementación

Docker puede ser descargado para el S.O Linux en la mayoría de distribuciones conocidas y Windows.

Asimismo Oracle ha certificado múltiples productos sobre Docker entre ellos: Oracle Database y Oracle Weblogic pero para ello debemos cumplir con ciertas especificaciones definidas.

Para esta implementación nos guiaremos de la siguiente matriz de certificación:

http://www.oracle.com/technetwork/middleware/ias/oracleas-supported-virtualization-089265.html

Asimismo recomiendo revisar las siguientes notas de Oracle Support:

- Oracle Weblogic 12.2.1 Support On Docker Multicontainer (Doc ID 2140342.1)
- Support for Oracle WebLogic Server Running in Docker Containers on Non-Certified Linux Host Operating Systems (Doc ID 2017945.1)

Realizaremos la instalación sobre Oracle Linux versión 7 Update 3 con el Kernel Unbreakable Enterprise, versiones inferiores no están soportadas. Los instaladores de Docker para dicha distribución están disponibles en el siguiente url: https://www.docker.com/docker-oracle-linux.

I. Instalación

La guía de instalación se encuentra documentado en el siguiente url:
https://docs.docker.com/engine/installation/linux/oracle/#os-requirements

En una instalación básica de Oracle Linux 7.3, los siguientes paquetes serán solicitados:

```
[root@srvdocker tmp]# rpm -ivh docker-ee-17.03.1.ee.3-1.el7.x86_64.rpm
warning: docker-ee-17.03.1.ee.3-1.el7.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID 76682bc9: NOKEY
error: Failed dependencies:
        docker-ee-selinux >= 17.03.1.ee.3-1.el7 is needed by docker-ee-17.03.1.ee.3-1.el7.x86_64
        libltdl.so.7()(64bit) is needed by docker-ee-17.03.1.ee.3-1.el7.x86_64
        libseccomp.so.2()(64bit) is needed by docker-ee-17.03.1.ee.3-1.el7.x86_64
        selinux-policy >= 3.13.1-102.0.3.el7_3.15 is needed by docker-ee-17.03.1.ee.3-1.el7.x86_64
```

Los paquetes marcados con color rojo son proporcionados por Docker al momento de descargar el producto.

Una vez instalado los pre-requisitos se procede a instalar Docker.

```
[root@srvdocker tmp]# rpm -ivh docker-ee-17.03.0.ee.1-1.el7.x86_64.rpm
warning: docker-ee-17.03.0.ee.1-1.el7.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID 76682bc9: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:docker-ee-17.03.0.ee.1-1.el7     ################################# [100%]
[root@srvdocker tmp]#
```

Nota: La distribución gratuita de Docker no está soportada para Oracle Linux. Se recomienda revisar el siguiente url: https://docs.docker.com/engine/installation/

Terminada la instalación, procedemos con la activación de los servicios.

Habilitando e Iniciando el Servicio de Docker:

```
[root@srvdocker ~]# systemctl enable docker
[root@srvdocker ~]# systemctl start docker
[root@srvdocker ~]# systemctl status docker
â docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service
   Active: active (running) since Sat 2017-05-13 09:23:00
     Docs: https://docs.docker.com
 Main PID: 1011 (dockerd)
   CGroup: /system.slice/docker.service
           ââ1011 /usr/bin/dockerd
           ââ1269 docker-containerd -l unix:///var/run/do

May 13 09:22:59 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:22:59 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:22:59 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:22:59 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:22:59 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:22:59 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:23:00 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:23:00 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:23:00 srvdocker.riccio.com dockerd[1011]: time=
May 13 09:23:00 srvdocker.riccio.com systemd[1]: Started
Hint: Some lines were ellipsized, use -l to show in full.
```

Nota: Recomiendo por temas de simplicidad detener el firewall incluido en Oracle Linux 7 con la finalidad de no tener puertos cerrados. Esto lo conseguimos con el comando: systemctl stop firewalld. Este comando debe ser ejecutado antes de subir los servicios de Docker.

II. Comandos Básicos

a) Validando la Versión de Docker:

```
[root@srvdocker ~]# docker version
Client:
 Version:      17.03.0-ee-1
 API version:  1.26
 Go version:   go1.7.5
 Git commit:   9094a76
 Built:        Wed Mar  1 01:15:13 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.03.0-ee-1
 API version:  1.26 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   9094a76
 Built:        Wed Mar  1 01:15:13 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

b) Información completa de la Instancia Docker:

```
[root@srvdocker ~]# docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 17.03.0-ee-1
Storage Driver: overlay2
 Backing Filesystem: xfs
 Supports d_type: false
 Native Overlay Diff: false
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 977c511eda0925a723debdc94d09459af49d082a
runc version: a01dafd48bc1c7cc12bdb01206f9fea7dd6feb70
init version: 949e6fa
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.1.12-61.1.18.el7uek.x86_64
Operating System: Oracle Linux Server 7.3
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 2.685 GiB
Name: srvdocker.riccio.com
ID: CUNH:KRY3:RME2:BJKX:TSMY:I26J:MT66:VK26:B23L:E6HB:ZTBV:QDQM
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
```

c) Listado de Imágenes:

```
[root@srvdocker ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

d) Listado de Containers:

```
[root@srvdocker ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

Nota: Todas los comandos están disponible a través de la ayuda: **docker --help**.

III. Creando Imagen

Una imagen en Docker, es una plantilla que cuenta con una serie de softwares instalados y que servirá para instanciar Containers. Los Containers harán copias de todos los archivos que irán modificando en el tiempo. Es importante notar que la imagen es estática y no será alterado por los Containers que la instancien.

Para crear la imagen que nos permitirá desplegar un Oracle HTTP Server 12c debemos seguir los siguientes pasos:

a) Creamos una carpeta.

```
[root@srvdocker ~]# cd $HOME
[root@srvdocker ~]# mkdir ohs
```

b) El archivo oraclelinux-7.3-rootfs.tar.xz complementará a nuestra imagen con una serie de comandos básicos de Linux. Dicho archivo se encuentra en el repositorio de Oracle:

https://github.com/oracle/docker-images/tree/OracleLinux-images/OracleLinux/7.3

El archivo lo copiamos en la carpeta previamente creada.

c) Descargamos la versión de Java 1.8 del siguiente url:
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Procedemos a copiar el instalador descomprimido en la carpeta previamente mencionada.

d) Copiar los siguientes archivos personalizados en el directorio ohs.

- Dominio que queremos desplegar en Docker. En mi caso a través del utilitario pack ($OH/ oracle_common/common/bin) generé el archivo web_friccio.jar que contiene el dominio que deseo desplegar.
- Generar un archivo Response File (rsp) de la Instalación de Oracle HTTP Server para ejecutarlo en modo no asistido.
- Crear el archivo oraInst.loc con el siguiente contenido:

```
inventory_loc=/u01/app/oraInventory
inst_group=oinstall
```

- Generar archivos que nos permitan iniciar el servicio de NodeManager y de OHS.
  En mi implementación he creado 2 archivos para este fin.

  - startall.sh

```
sh
/u01/app/oracle/product/middleware/Oracle_Home_1/user_projects/domains/web_friccio/bin/startNodeManager.sh &
sleep 5
sh
/u01/app/oracle/product/middleware/Oracle_Home_1/oracle_common/common/bin/wlst.sh /u01/start-ohs.py
echo "EJECUTÁNDOSE - OHS"
read
```

  - start-ohs.py (Utilizando WLST).

```
import os, sys
java_home=os.environ.get("JAVA_HOME","/usr/java/jdk1.8.0_131")
nm_pass=os.environ.get("NM_PASSWORD","oraclesqldb2")
ohs_comp_name=os.environ.get("OHS_COMPONENT_NAME","ohs1")
domain_name=os.environ.get("DOMAIN_NAME","web_friccio")
domain_path='/u01/oracle/ohssa/user_projects/domains/' + domain_name
oracle_home =
os.environ.get("ORACLE_HOME","/u01/app/oracle/product/middleware/Oracle_Home_1")
nmConnect(username='weblogic',password=nm_pass,domainName=domain_name)
nmServerStatus(serverName=ohs_comp_name,serverType='OHS')
nmStart(serverName=ohs_comp_name,serverType='OHS')
nmServerStatus(serverName=ohs_comp_name,serverType='OHS')
exit()
```

e) Identificar los rpms requeridos para certificar el Sistema Operativo y los copiamos en una sub-carpeta de ohs.

Después de estos puntos, tendríamos los siguientes archivos:

```
[root@srvdocker ohs]# ls -lrt
total 901736
-rwxrwxr-x. 1 root root 703518462 Oct 10  2016 fmw_12.2.1.2.0_ohs_linux64.bin
-rwxrwxr-x. 1 root root  49130232 Apr 21 02:21 oraclelinux-7.3-rootfs.tar.xz
-rwxrwxr-x. 1 root root      2084 May 13 13:19 ohs.rsp
-rwxrwxr-x. 1 root root 169983496 May 13 13:25 jdk-8u131-linux-x64.rpm
-rwxrwxr-x. 1 root root        56 May 13 19:18 oraInst.loc
-rwxrwxr-x. 1 root root       654 May 13 21:28 start-ohs.py
drwxrwxr-x. 2 root root      4096 May 13 22:46 rpm
-rwxrwxr-x. 1 root root    714461 May 14 00:23 web_friccio.jar
-rwxrwxr-x. 1 root root       244 May 14 02:25 startall.sh
-rwxrwxr-x. 1 root root      1595 May 14 02:56 Dockerfile
```

f) Creamos el archivo de configuración de la imagen, para ello creamos un archivo con el nombre Dockerfile con el siguiente contenido:

```
1  FROM scratch
2  ADD oraclelinux-7.3-rootfs.tar.xz /
3  RUN mkdir -p /install/rpm
4  ADD jdk-8u131-linux-x64.rpm /install
5  ADD fmw_12.2.1.2.0_ohs_linux64.bin /install
6  ADD oraInst.loc /install
7  ADD ohs.rsp /install
8  ADD web_friccio.jar /install
9  ADD startall.sh /install
10 COPY rpm/* /install/rpm/
11 WORKDIR /install/rpm
12 RUN rpm -ivh libaio-0.3.106-5.x86_64.rpm
13 RUN rpm -ivh libaio-devel-0.3.106-5.x86_64.rpm
14 RUN rpm -ivh kernel-headers-2.6.18-371.el5.x86_64.rpm
15 RUN rpm -ivh libgomp-4.4.7-1.el5.x86_64.rpm
16 RUN rpm -ivh cpp-4.1.2-54.el5.x86_64.rpm
17 WORKDIR /install
18 RUN chmod 775 *
19 RUN mkdir -p /usr/share/man/man1/
20 RUN chmod 775 jdk-8u131-linux-x64.rpm; rpm -ivh jdk-8u131-linux-x64.rpm; java -version
21 RUN groupadd oinstall
22 RUN useradd -g oinstall -d /home/oracle oracle
23 RUN echo "riccio" | passwd "oracle" --stdin
24 RUN mkdir /u01
25 RUN cp /install/oraInst.loc /etc; chmod 775 /etc/oraInst.loc
26 RUN chown oracle.oinstall /install; chown oracle.oinstall /u01
27 RUN cp /install/startall.sh /u01/.; chown oracle.oinstall /u01/startall.sh
28 USER oracle
29 WORKDIR /install
30 ENV OHS_INSTALL=/install/fmw_12.2.1.2.0_ohs_linux64.bin \
31     RSP=/install/ohs.rsp \
32     JAVA_HOME=/usr/java/jdk1.8.0_131
33 RUN $OHS_INSTALL -ignoreSysPrereqs -novalidation -silent -responseFile $RSP -jreLoc $JAVA_HOME
34 WORKDIR /u01/app/oracle/product/middleware/Oracle_Home_1/oracle_common/common/bin
35 RUN sh unpack.sh \
36     domain=/u01/app/oracle/product/middleware/Oracle_Home_1/user_projects/domains/web_friccio \
37     -template=/install/web_friccio.jar
38 ADD start-ohs.py /u01
39 WORKDIR /u01
40 RUN chmod 775 /u01/startall.sh
41 EXPOSE 7777
42 CMD ["/bin/sh","/u01/startall.sh"]
```

Donde:

LÍNEA	EXPLICACIÓN
1	Toda imagen debe basarse en una imagen y en este caso la imagen SCRATCH es la imagen más básica que presenta Dockers.
2	Copiamos desde nuestro servidor el archivo oraclelinux-rootfs.tar.xz al entorno virtualizado en Dockers en la raíz del Sistema Operativo.
3	El comando RUN nos permite ejecutar un comando en el entorno virtualizado. En este caso creamos dos carpetas: /install y /install/rpm.
4-9	Copiamos todos los archivos mencionados en los puntos previos en las respectivas carpetas.
10	El comando COPY es similar al comando RUN con la única diferencia que si existe un archivo comprimido lo descomprime si es conocido el formato.

11	El comando WORKDIR permite ubicarnos en una carpeta específica para ejecutar algún comando posteriormente.
12-16	Procedemos a instalar todos los paquetes requeridos para contar con el Sistema Operativo certificado. Por temas didácticos he instalado solo los principales. Recomiendo configurar el repositorio YUM para simplificar la instalación de los pre-requisitos.
17-20	Procedemos a instalar JAVA 1.8 en el entorno virtualizado por Docker.
21-23	Creamos el grupo oinstall y el usuario oracle con el password “riccio”. Estos pasos son requeridos para realizar posteriormente la instalación de Oracle HTTP Server 12c.
24-26	Definimos el oraInventory y entregamos los accesos adecuados.
27 y 38	Copiamos los archivos que nos permitirán iniciar el NodeManager y el OHS.
28	Nos logueamos con el usuario oracle.
29-33	Procedemos a definir las variables de ambiente e instalamos Oracle HTTP Server 12c de manera no asistida.
34-37	Procedemos a copiar el Dominio en el Oracle_Home previamente desplegado.
39-40 y 42	Procedemos a entregar los correctos permisos a los archivos que permitirán subir los servicios.
41	Procedemos a publicar los puertos que deseamos que el servidor anfitrión pueda tener acceso, en este caso sería el puerto 7777.
42	Procedemos a ejecutar los scripts que permiten iniciar los servicios NodeManager y OHS. La diferencia entre el comando RUN y CMD es que el primero se ejecuta durante la fase de BUILD, el segundo se ejecuta en la fase de RUN. Adicionalmente el comando CMD solo puede ser especificado 1 vez, si hay más los omitirá.

e) Construimos la imagen.

docker build --force-rm=true -t "ohs/friccio:v1" .

Donde el parámetro –t indica el nombre y tag adicional de la imagen.

Al ejecutar el comando deberíamos tener una salida como se presenta:

```
Sending build context to Docker daemon 1.036 GB
Step 1/41 : FROM scratch
 --->
Step 2/41 : ADD oraclelinux-7.3-rootfs.tar.xz /
 ---> Using cache
 ---> 3ad20c1a1acc
Step 3/41 : RUN mkdir -p /install/rpm
 ---> Using cache
 ---> ee18047256c4
Step 4/41 : ADD jdk-8u131-linux-x64.rpm /install
 ---> Using cache
 ---> b54c06848d47
```

Posterior a la creación deberíamos visualizarlo:

```
[root@srvdocker ohs]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ohs/friccio         v1                  b3482cd31038        37 minutes ago      4.05 GB
```

Cabe mencionar que la imagen se almacenará en la ruta: /var/lib/Docker como veremos a continuación:

```
[root@srvdocker docker]# pwd
/var/lib/docker
[root@srvdocker docker]# ls -l
total 4
drwx------. 2 root root    6 May 13 11:48 containers
drwx------. 3 root root   21 May 11 00:46 image
drwxr-x---. 3 root root   18 May 11 00:46 network
drwx------. 4 root root 4096 May 13 11:48 overlay2
drwx------. 4 root root   30 May 11 00:46 plugins
drwx------. 2 root root    6 May 11 00:46 swarm
drwx------. 2 root root    6 May 13 11:48 tmp
drwx------. 2 root root    6 May 11 00:46 trust
drwx------. 2 root root   24 May 11 00:46 volumes
[root@srvdocker docker]# cd image
[root@srvdocker image]# ls -l
total 0
drwx------. 5 root root 77 May 13 11:48 overlay2
[root@srvdocker image]# cd overlay2/
[root@srvdocker overlay2]# ls -l
total 4
drwx------. 2 root root   6 May 11 00:46 distribution
drwx------. 4 root root  35 May 11 00:46 imagedb
drwx------. 5 root root  42 May 13 11:48 layerdb
-rw-------. 1 root root 129 May 13 11:48 repositories.json
[root@srvdocker overlay2]# cat repositories.json
{"Repositories":{"ohs/friccio":{"ohs/friccio:latest":"sha256
```

Nota: Si en algún momento tenemos problemas con el punto de montaje overlay2 simplemente procedemos a ejecutar los siguientes pasos para recrearlo:

- Desmontar el punto de montaje (umount / var/lib/docker/overlay2).
- rm –Rf /var/lib/docker
- systemctl restart docker.

IV. Creando Container

Un Container en Docker es la instancia de una imagen, donde una imagen podría tener n contenedores y cada uno de ellos es independiente.

Para crear y ejecutar un container ejecutamos cualquiera de los siguientes comandos:

- docker run -i -t ohs/friccio:v1 (Inicia en modo Asistido)
- docker run -d -t ohs/friccio:v1 (Inicia en modo Background)

Para visualizar los containers en ejecución ejecutamos lo siguiente:

```
[root@srvdocker ohs]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
73e1e68af87a        ohs/friccio:v1      "/bin/sh /u01/star..."   36 minutes ago      Up 36 minutes
```

Para nuestra implementación he agregado los siguientes parámetros adicionales:

- -m 6G => Definí un límite de 8 GB (Incluyendo la memoria SWAP) para el ambiente virtualizado. Si deseamos excluir la memoria SWAP de manera independiente podemos utilizar el parámetro --memory_swap).

- -p 7778:7777 => Aquí permitimos que el servidor anfitrión pueda publicar por el puerto 7778 el contenido que está presentando el ambiente virtualizado en Docker por el puerto 7777. Si también necesitamos definir por una ip específica, podemos configurarlo de la siguiente manera: -p IP:PUERTO_ANFITRION:PUERTO_VIRTUAL. Asimismo si necesitamos definir el protocolo UDP, simplemente agregamos al final /udp (7778:7777/udp).

Algunas operaciones sobre Containers que nos pueden servir:

- docker start –t “<CONTAINER_ID>” => Inicia un Container previamente creado.

- docker stop –t “<CONTAINER_ID>” => Detiene un Container.

- docker rm –t “<CONTAINER_ID>” => Elimina un Container.

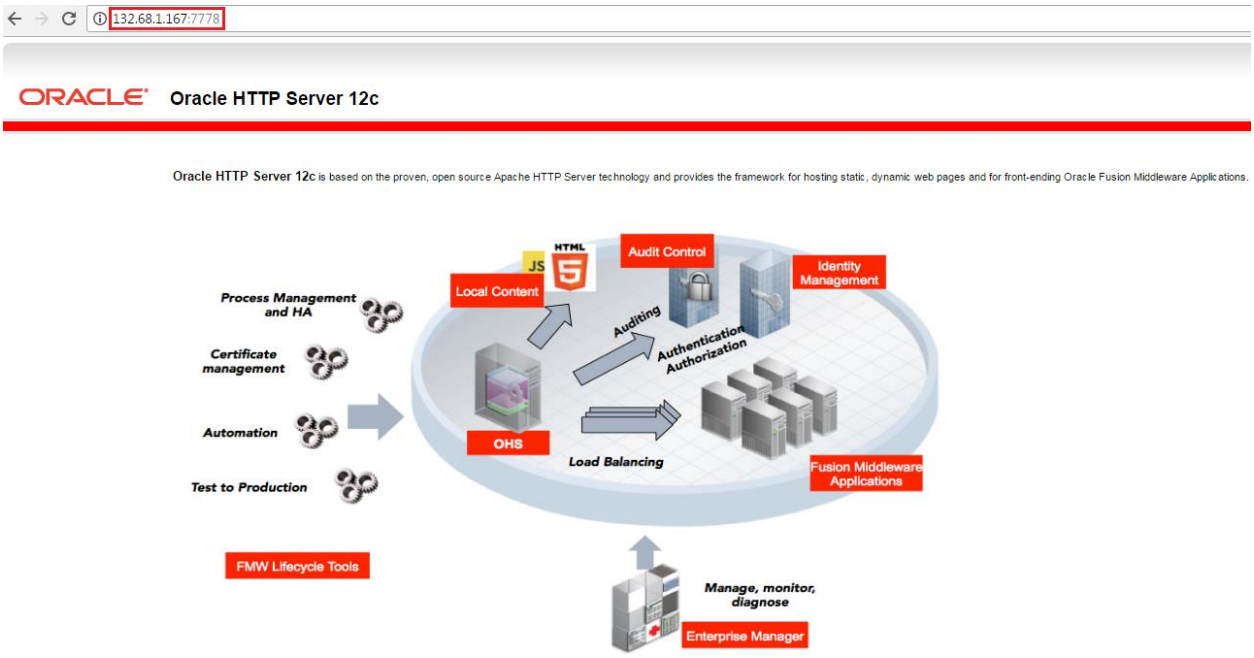
- docker rmi –t “<CONTAINER_ID>” => Elimina una Imagen.

- docker exec -t “<CONTAINER_ID>” <Comando de S.O> => Permite ejecutar comandos a nivel de S.O a un Container en ejecución, ejemplo:

```
[root@srvdocker ohs]# docker exec -t "73e1e68af87a" ps -ef
UID         PID   PPID  C STIME TTY          TIME CMD
oracle        1      0  0 08:01 ?        00:00:00 /bin/sh /u01/startall.sh
oracle        5      1  0 08:01 ?        00:00:00 sh /u01/app/oracle/product/middl
oracle        7      5  0 08:01 ?        00:00:00 /bin/sh /u01/app/oracle/product/
oracle       53      7  0 08:01 ?        00:00:06 /u01/app/oracle/product/middlewa
oracle      132      1  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      133    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      134    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      135    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      136    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      137    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      138    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      139    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      140    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle      236      1  0 08:01 ?        00:00:00 /bin/bash
oracle      315    132  0 08:01 ?        00:00:00 /u01/app/oracle/product/middlewa
oracle     2452      0  0 08:19 ?        00:00:00 ps -ef
```

  Aquí estamos mostrando todos los procesos que están ejecutándose que pertenecen al usuario Oracle dentro del entorno virtualizado.

Por último procedemos a probar la página web por defecto que se encuentra en Oracle HTTP Server. Para ello, nos conectaremos a la IP del servidor Oracle Linux 7.3 a través del puerto 7778, donde este, está haciendo NAT al puerto 7777 del entorno virtual en Docker.



Conclusión

Como podemos apreciar, hoy tenemos una tecnología disponible desde hace unos años que nos permite ir a otro nivel de virtualización distinto, permitiéndonos obtener las siguientes ventajas:

- Instalación simple y capacidad de ejecutar múltiples aplicaciones en entornos aislados sobre un mismo sistema operativo, permitiéndonos ahorrar horas de trabajo en la administración de Infraestructura.
- Independiente a la plataforma, permite contar con soluciones más portables.
- Despliegue de Aplicaciones mucho más rápida y flexible.
- Disponible en múltiples proveedores de Nube.

Asimismo es importante siempre validar la matriz de certificación en caso exista del producto a desplegar con la finalidad de contar con una solución estable y con soporte.

Publicado por Ing. Francisco Riccio. Es un Cloud Architect en IBM Perú e instructor de cursos oficiales de certificación Oracle. Está reconocido por Oracle como un Oracle ACE y certificado en productos de Oracle Application & Base de Datos.

e-mail: franciscoriccio@gmail.com

web: www.friccio.com