Sistemas Operativos
Trabajo Práctico Final
Docker©

Cátedra:
Profesor: Lic. Marcelo Gómez

Ayudantes de cátedra:
Lic. Lucy Marticorena
APU Leandro Luque

Alumnos:
SERRUYA ALOISI, Luciano
TOLEDO MARGALEF, Pablo

4 de julio de 2017

Índice

1. Introducción **2**
1.1. Máquinas virtuales 3
1.2. Plataformas de contenedores 4

2. ¿Qué es Docker©? **7**
2.1. ¿Qué **NO** es Docker©? 7
2.2. Arquitectura 7

3. Ventajas y desventajas **8**

4. Ejemplo práctico **9**

1. Introducción

Un buen punto de partida antes de empezar a describir la tecnología Docker©, sería hacer una breve introducción a la **virtualización**.

El sitio de RedHat Inc. describe a la virtualización de la siguiente manera:

> La virtualización es una tecnología que permite crear múltiples ambientes simulados o recursos dedicados a partir de un sistema de hardware. Un software llamado *hypervisor* se conecta directamente con ese hardware y brinda la posibilidad de dividir un sistema en varios entornos separados, distintos y seguros conocidos como **Máquinas virtuales**. Dichas máquinas se basan el la habilidad del *hypervisor* para abstraer los recursos del hardware y distribuirlos entre las máquinas virtuales acordemente [1]

Las tecnologías que permiten la virtualización surgieron en la década del 1960, sin embargo obtuvieron mayor popularidad cuando varias empresas tenían que correr software de distintos fabricantes en máquinas de un fabricante en particular (que sólo permitían correr su software).

Utilizando la virtualización, dichas organizaciones podían correr software de distintos fabricantes usando distintos tipos y versiones de sistemas operativos. De esta forma, los servidores se utilizaban más eficientemente, reduciendo costos de compras, de instalación, mantenimiento, y refrigeración.

Los recursos que comunmente se virtualizan son los **servidores**, **sistemas operativos**, y las **redes**.

Virtualizar servidores sirve para maximizar su rendimiento, logrando así poder atender más solicitudes de clientes y poder usar sus componentes para realizar más funciones. Virtualizar sistemas operativos permite tener distintos sistemas operativos corriendo a la vez

(por ejemplo, tener una imagen de Windows, y otra de alguna distribución Linux). La virtualización de redes reduce los componentes físicos necesarios para crear múltiples redes independientes entre sí.

1.1. Máquinas virtuales

Como se describió anteriormente, las máquinas virtuales son una **abstracción de los recursos del hardware**. Son programas que emulan una computadora, dando la sensación de que lo que ejecuta la máquina virtual lo ejecuta directamente sobre el hardware (por ejemplo, la máquina virtual cree que tiene un disco duro de 10GB, mientras que realmente es un archivo más en el filesystem del sistema operativo anfitrión).

La capa intermedia que existe entre la máquina virtual y el hardware real se encarga de crear y administrar los recursos físicos para las distintas instancias de máquinas virtuales que estén corriendo en simultáneo. Esta capa intermedia puede correr encima del sistema operativo anfitrión (como por ejemplo VirtualBox, también llamados *Type 2 Hypervisors*), o se puede ejecutar directamente sobre el hardware (*Type 1 Hypervisors*, por ejemplo KVM)[3]. Esta última opción logra mejores rendimientos que la primera ya que se elimina una capa intermedia (que sería el sistema operativo anfitrión), siendo el *hypervisor* el anfitrión, y las máquinas virtuales corren encima de él.

Esta opción de virtualización consume bastantes recursos de la máquina anfitriona, ya que debe emular el sistema completo, desde los recursos físicos (disco duro, procesador, RAM), hasta el filesystem del sistema operativo huésped.

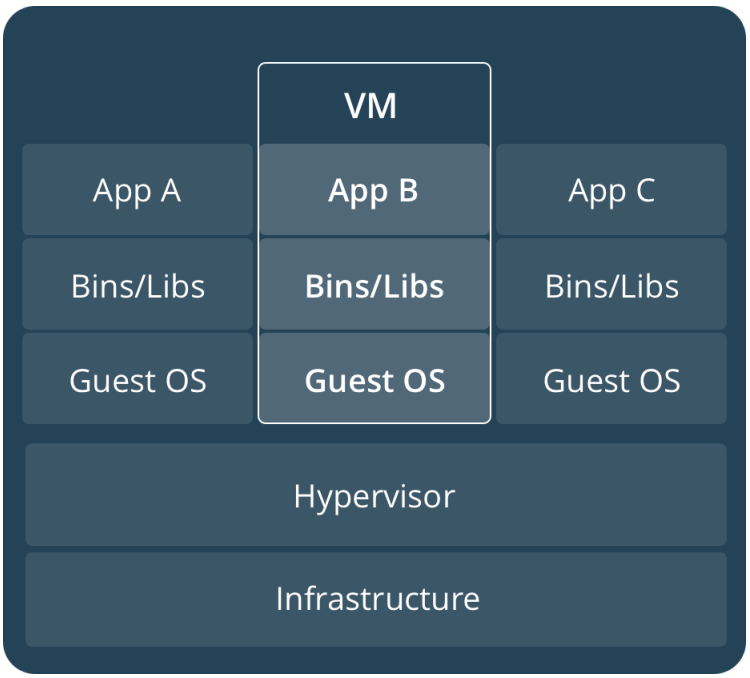

Figura 1: Máquinas virtuales sobre *hypervisor*

1.2. Plataformas de contenedores

Por otro lado existen también los **contenedores**, que el sitio de Docker© los describe como "paquete ejecutable liviano y autónomo de una pieza de software que incluye todo lo que necesita para correr: código, librería del sistema, configuraciones".

Los contenedores son una abstracción en la capa de la aplicación que aislan el software que se desea ejecutar, armando un pequeño ambiente con las dependencias necesarias para que se ejecute solamente ese código. Esto facilita el trabajo de los desarrolladores, sabiendo que al trabajar y probar su producto en un contenedor, ese mismo contenedor podrá ser puesto en producción con la certeza de que funcionará, evitando incertidumbres que pueden generar intentar de ejecutar el software en distintos sistemas operativos o hardware.

El sistema operativo anfitrión sobre el que corren los contenedores restringe el acceso del contenedor a los recursos físicos, por lo tanto un solo contenedor no podrá consumir todos los recursos

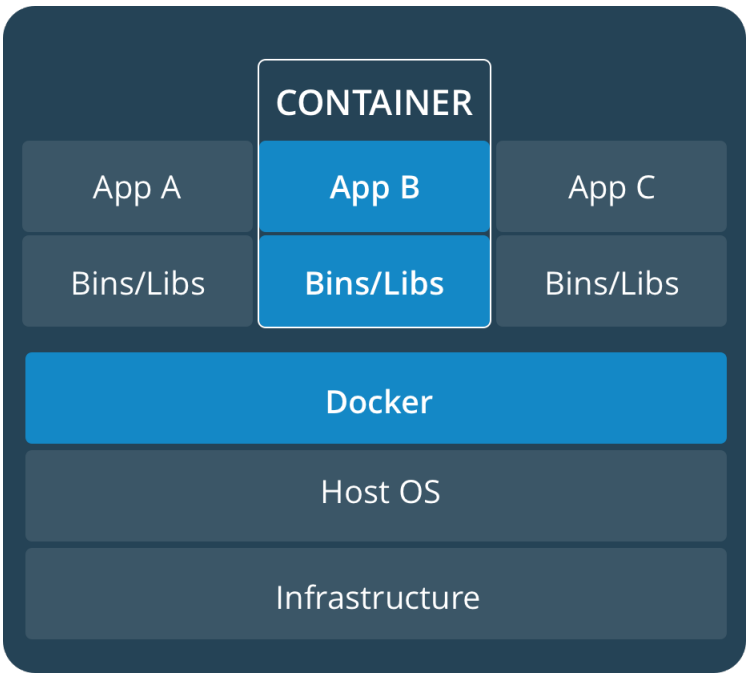

Figura 2: Contenedores corriendo sobre la plataforma Docker©

del anfitrión. Como todos los contenedores comparten el kérnel del sistema operativo anfitrión, logran que su uso (también llamado *virtualización basada en contenedores*) sea más eficiente que el uso de las máquinas virtuales (donde cada máquina virtual ejecutaba un sistema operativo entero)

> La diferencia clave entre los contenedores y las máquinas virtuales es que, mientras el *hypervisor* abstrae un dispositivo entero, los contenedores solo abstraen el kérnel del sistema operativo. [4]

Al momento de momento de decidir si usar una máquina virtual o un contenedor para ejecutar una aplicación de forma aislada, no se debe perder de vista el **alcance** que se espera que tenga la aplicación.[2]

Al ser los contenedores mucho más livianos que las máquinas virtuales (no tienen que correr una instancia del sistema operativo,

sólo la aplicación para la cual fueron diseñados), permiten elevar más el nivel de abstracción, logrando tener un contenedor altamente específico para correr una aplicación en particular. Como práctica general, se mantiene el concepto de tener **un proceso por contenedor**.

Por otro lado, las máquinas virtuales tienen un alcance mucho más amplio, permitiendo correr sistemas operativos enteros.

2. ¿Qué es Docker©?

2.1. ¿Qué NO es Docker©?

2.2. Arquitectura

hablar sobre lo que dice el informe, también hablar sobre registry (permite armar un repo local, y los clientes pueden descargar de ahí)

3. Ventajas y desventajas

4. Ejemplo práctico

Referencias

[1] URL https://www.redhat.com/es/topics/virtualization.

[2] Scott Lowe. Virtual machines vs. containers: A matter of scope, May 2014. URL http://www.networkcomputing.com/cloud-infrastructure/virtual-machines-vs-containers-matter-scope/2039932943.

[3] Bhanu Tholeti. Learn about hypervisors, system virtualization, and how it works in a cloud environment, September 2011. URL https://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/.

[4] Steven J. Vaughan-Nichols. What is docker and why is it so darn popular?, May 2017. URL http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/.