Implantación de Tecnologías de Contenedores en una Organización

Autor: Alonso Batuecas, Francisco

Director/es: Suárez Lorenzo, Fernando y Fernández García, Norberto**.**

Contacto: fab@interior.es

Resumen: El contenido de este Trabajo de Fin de Máster, versa sobre la implantación de tecnologías tipo contenedores, dentro de las infraestructuras de sistemas y comunicaciones de una organización, desde el punto de vista de un Director de Infraestructura con la experiencia de años en este campo y contextualizada en la fecha de realización del mismo.

Se pretende describir en qué consiste este tipo de tecnología, profundizando en los diferentes componentes que la forman, cómo se relacionan entre sí, qué funciones realizan y sobre qué tipos de servidores y sistemas de comunicaciones se pueden desplegar.

Analizar las diferentes posibilidades existentes a la hora de su implantación, productos comerciales y de software libre, así como abordar su despliegue, necesidades y consideraciones a tener en cuenta.

Por otro lado, se trata de introducir la metodología *DevOps* y su integración dentro de la infraestructura *dockerizada*, exponiendo en qué consiste la integración continua.

Por último, se pretende abordar la *securización* o retos de seguridad en estas plataformas y las claves para poder aplicar buenas prácticas de cara a una configuración segura de la plataforma.

Para finalizar, se presentan las conclusiones en función de todo lo desarrollado durante el TFM, que permitan tomar una decisión a la hora de abordar una implantación o cambio tecnológico dentro de una organización, en función de las diferentes posibilidades existentes.

.

Palabras clave: Docker, Infraestructura, Administración, Seguridad, Desarrollo, Contenedores

1. Introducción

Los “Contenedores” no son un concepto nuevo, los primeros pasos se dieron sobre el sistema operativo Unix unos pocos años antes del 1979, año en el que comienza a darse sobre los sistemas operativos Linux. Desde entonces, esta tecnología no ha parado de evolucionar, para llegar a la

situación actual que se prevé que no sea la final y siga con la constante evolución, a continuación se muestra en la Figura 2-1 una línea de tiempo de dicha evolución. [1]

Figura 2-1 Historia contenedores (Fuente: [1]**)**

En los inicios, los esfuerzos sobre esta tecnología se centraban en mejorar el aislamiento de los contendores, lo que dio origen a "*Solaris Containers*", tecnología que aprovechó la función mejorada del «Sistema Operativo Solaris» llamada "*Zones*" para aislar aún más los contenedores.

Como siguiente hito, destacan las contribuciones de Google que aportaron como resultado la creación de varias arquitecturas de contenedores en el año 2006. Estas tenían la capacidad de particionar y asignar recursos de hardware, almacenamiento y red a los contenedores creados, proporcionando a los gestores mayor control y granularidad sobre los mismos y a su vez, tener mayor visibilidad sobre cómo podían influir en la seguridad.

Estas mejoras, poco a poco, se fueron incorporando en los sistemas operativos Linux, lo cual impulsó la evolución de las Tecnologías de Virtualización basada en Contenedores conocidas actualmente. Destacándose las conocidas como *LXC*, *LMCTFY* de Google, que luego derivaron en *Kubernetes*, y *Docker*.

1.1. Introducción a la problemática

Para poder entender mejor, por qué se da el salto a una arquitectura de contendores, tenemos que conocer cuál es el estado del arte en el desarrollo de software y la gestión de la infraestructura en la cual se despliegan esos desarrollos.

Los retos o problemas a los que nos enfrentamos los podemos clasificar, en función de la fase en la que se producen, en tres categorías:

- Problemas durante la construcción del software.
  - Dependencias de desarrollo.
  - Versiones de entornos de ejecución.
  - Equivalencia de entornos de desarrollo.
  - Equivalencia de entornos de producción.

    - Versiones / compatibilidad con terceras partes.
- Problemas durante la distribución del software.
    - Generaciones de entregables (builds) diferentes.
    - Acceso a servidores de producción.
    - Ejecución nativa vs. Ejecución distribuida.
- Problemas para la ejecución del software.
    - Dependencias de aplicación
    - Compatibilidad de Sistema Operativo
    - Disponibilidad de servicios externos
    - Recursos hardware

1.2. Soluciones adoptadas

Se suele abordar la respuesta a estos problemas, con entornos pre-productivos, intentando mantenerlos nivelados con los entornos de producción. Haciendo uso de metodologías y procedimientos para el desarrollo de software y despliegue de aplicaciones, o utilizando las ventajas de la virtualización para conseguir distribuir entornos y acercarlos al desarrollador con la idea de atajar los problemas mencionados.

Con la virtualización, se pasó a una sencillez y flexibilidad que antes era impensable, simplemente con una platilla bien configurada era posible replicar servidores y crear un entorno pre-productivo igual que el productivo, facilitando la labor a los responsables de infraestructura.

Pero el mundo de la tecnología no para de evolucionar y fijándose en el sector del transporte y las mercancías, parece que ha encontrado una solución muy optimizada para los problemas mencionados. A la fecha de redacción de este trabajo el estado del arte pasa por la infraestructura de contenedores y los desarrollos orientados a micro-servicios que aprovechen las bondades de dichas plataformas, tal como se mencionan en numerosos artículos especializados [2]. [3]

Ahora, extrapolado al mundo tecnológico, las soluciones basadas en contenedores, lo que prometen es que se pueda construir, distribuir y ejecutar el código desarrollado en cualquier lugar sin tener problemas del entorno en el cual se ejecutan, dentro de éstas, Docker constituye una de las más extendidas en la industria.

Solomon Hykes comenzó Docker como un proyecto interno dentro dotCloud, empresa enfocada a una plataforma como servicio (PaaS), con las contribuciones iniciales de otros ingenieros de 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 es a su vez construida sobre proyectos de código abierto anteriores como Cloudlets [4].

Desde entonces hasta ahora Docker, ha conseguido ser el referente en cuanto a plataformas de contenedores se refiere, aunque le están surgiendo otros competidores como Kubernetes u OpenShift.

Kubernetes (K8s) que nació en 2014 como una plataforma de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones en contenedores. Kubernetes agrupa los contenedores que conforman una aplicación en unidades lógicas para una fácil administración y descubrimiento. [5].

Red Hat OpenShift es la plataforma de nube híbrida con el respaldo de la empresa Red Hat: que para infraestructuras empresariales cuenta con el soporte del propio Red Hat, ofreciendo características similares al resto [6].

En la figura 2-2, se puede ver la evolución del uso de la solución *Docker*, y como en los años 2016 y 2017, experimenta una subida exponencial.

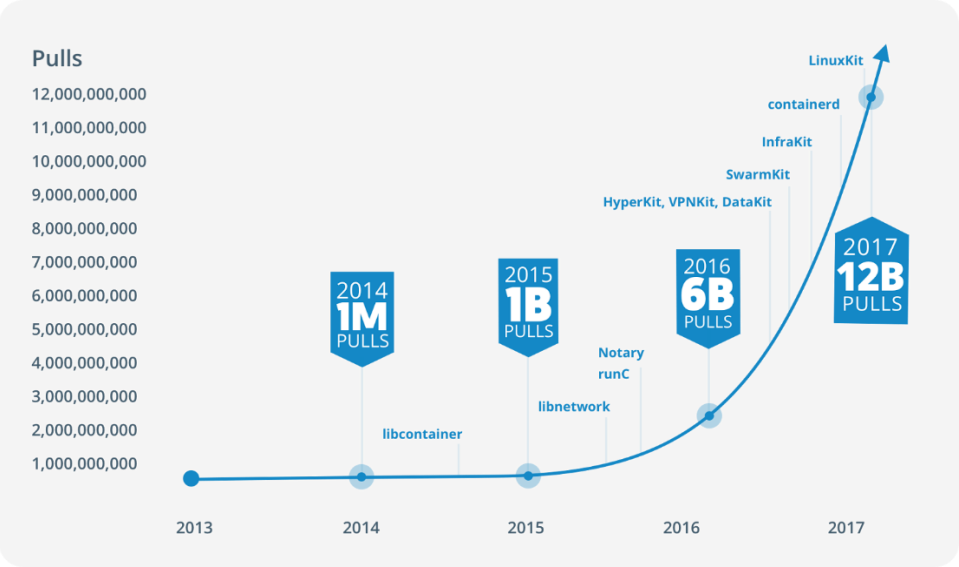

Figura 2-2 Evolución contenedores (Fuente: [7]**)**

2. Desarrollo

La Containarization, también conocida como virtualización basada en contenedores, es una modalidad de virtualización a nivel de Sistema Operativo, que nos permite desplegar y ejecutar aplicaciones sin necesidad de contar con una máquina virtual completa, con su S.O y asignación de recursos.

En su lugar, se puede montar uno o varios sistemas aislados denominados contenedores. Estos sistemas pueden ser ejecutados sobre un único servidor huésped host y acceden al mismo núcleo (kernel), que el de su servidor alojador, por lo cual comparten el mismo S.O sin necesidad de montar uno para cada contenedor, a diferencia de las máquinas virtuales, que además requieren de un hipervisor para la gestión de los recursos del host, que consume recursos del mismo y conforman una infraestructura más pesada que la de contenedores.

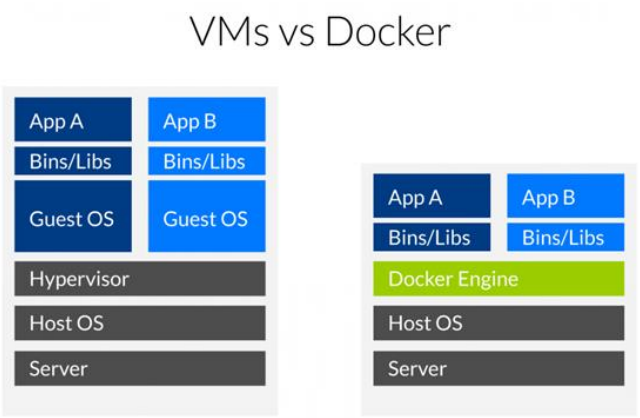

Figura 3 1 VMs vs Docker (Fuente: [8] **)**

En la siguiente tabla, se pueden ver las características de la Containarizacion y la virtualización, al objeto de poder comprender de una forma más clara las diferencias entre estas dos tecnologías.

Containarization	**Virtualización**
Virtualización basada en SO Múltiples contenedores sobre el mismo OS	**Virtualización basada en hardware** Múltiples SO comparten los recursos del *host*
Ligero Servicios de aplicaciones compartiendo recursos del SO	**Pesado** SO adicionales gestionando recursos para cada *guest*
Aprovisionamiento en tiempo real De forma sencilla, en el momento en el que se comienzan a ejecutar nuevos servicios de aplicaciones	**Aprovisionamiento lento** Requiere inicialización del *guest*, más el tiempo de arranque del SO
Ejecución nativa Acceso directo a los recursos del hardware	**Ejecución limitada** Acceso a los recursos del hardware por medio de la capa de virtualización
Menos seguridad Aislamiento únicamente a nivel de procesos	**Alta seguridad** Aislamiento total

Tabla 1. Características de Containarization y Virtualización

2.1. Contenedor

El contenedor es la pieza fundamental sobre la que se apoya la infraestructura de contenedores, básicamente es un proceso o agrupación de procesos, en función de la complejidad del mismo, que solo pueden ejecutarse en el propio contexto del contenedor. Es decir, los procesos están aislados y solamente pueden usar los recursos definidos en el contenedor.

En el caso de *Docker*, los procesos corren de forma nativa en máquinas Linux, lo único que se comparte es el *kernel*, por eso en ambientes productivos se usa *Docker* instalado sobre Linux.

Para ello Linux aísla los contenedores utilizando los conceptos de *Chroot*, *Cgroups* y *Namespaces.*

2.2. Docker

Docker surge como un proyecto de código abierto basado en contenedores de Linux. Se puede definir como un "motor de contenedores", para ello utiliza las características del núcleo de Linux que se han explicado en el apartado anterior, permitiéndole crear los contenedores por encima del S.O. sin necesidad de montar un S.O. por cada contenedor, a diferencia de las M.V.

A continuación, se muestra un resumen de los componentes más importantes de *Docker*

Componente	**Función**
Servidor	Proceso lanzado mediante el comando, "dockerd"
Rest API	Interfaz de comunicación entre los programas y el demonio de Docker

cliente	Consola de línea de comandos (*CLI*)
Imágenes de *Docker* (Docker Images)	Plantilla en modo solo lectura, con toda la configuración de un contenedor en el fichero de manifiesto (*Docker file*)
Registros de *Docker* (Docker Registries)	Componente utilizado para la distribución, define los repositorios de imágenes.
Contenedores de *Docker* (Docker Containers)	Componente con todo lo necesario para ejecutar las aplicaciones o servicios
Docker Swarm	Componente para la creación y gestión de *Clusters*
Docker Machine	Herramienta para el despliegue de contenedores en servidores virtuales
Docker Compose	Herramienta de gestión en infraestructuras con elevado número de contenedores

Tabla 2. Resumen Componentes *Docker*.

2.3. Aspectos de Seguridad en Docker

A continuación, se muestran recomendaciones para bastionar nuestra infraestructura de contenedores, en función del componente que estemos configurando:

Componente	**Función**
Host	Chequear la partición de Instalación Gestionar los permisos de los usuarios con gestión sobre la plataforma Auditar los ficheros y directorios de la instalación de *Docker*
Docker Daemon	Limitar el tráfico entre contendores Verificar la autorización de uso de la CLI Realizar una buena gestión de registros (*logs*) Configurar los permisos de acceso a los ficheros del Daemon
Imágenes de *Docker* (Docker Images)	Revisar los permisos de ejecución (root) Uso del Content Trust para revisar la integridad de las imágenes Vigilar que no queden almacenados secretos
Contenedores de *Docker* (Docker Containers)	Restringir los protocolos de uso Configurar una política de limitación de uso de recursos

Tabla 3. Buenas prácticas de seguridad.

3. Resultados y discusión

Como objetivos para la realización de este trabajo, se planteaban identificar las cuestiones que un director de infraestructura debe tener en cuenta, a la hora de abordar la implantación de una plataforma basada en contendores, por lo que a continuación se muestran los resultados obtenidos:

- Perfil de organización: No siempre encajan en nuestra organización, ni tampoco son sustitutivas de la virtualización, incluso pueden ser complementarias.
  Si el negocio tecnológico de la organización se limita al uso de los sistemas de información comerciales, puede ser que no sea el perfil ideal para dar el salto a los contendores, ya que en muchas ocasiones los paquetes comerciales de aplicaciones, no están desarrollados para ser desplegados sobre contendores y si sobre máquinas virtuales, en este caso la optimización que se persigue con una estrategia de contenedores es inviable.
  En el caso de que la organización se dedique al desarrollo de sistemas de información, y cuente con el conocimiento necesario, si es una opción, pero siempre y cuando los sistemas a desarrollar estén basados en micro-servicios, o productos atomizados, que son los que obtendrán mejores resultados en el despliegue en contendores.
- Elección de plataformas: Se han analizado durante el trabajo las plataformas Docker, OpenShift, Kubernetes y Rocket, pero no son las únicas, ya que también existen plataformas de contendores como servicio (CaaS), donde se consume el servicio desde una nube especializada en esta materia.
  En función de las preferencias de la organización y el tipo de sensibilidad de la información que maneje, tocara abordar esta cuestión al igual que cualquier servicio de la nube, con las ventajas y desventajas de los mismos.
  Si la organización maneja información sensible, lo normal es que opte por plataformas en sus propias instalaciones.
- Tipo de implantación: En el caso de que el servicio exija una disponibilidad elevada, existen los *cluster* de contenedores, pero si estamos pensando en servicios 24x7x365, y con la posibilidad de un buen plan de continuidad de negocio apoyándose en otros CPD o en una infraestructura de nube hibrida, quizá sea interesante un planteamiento tomando como base la virtualización con un hipervisor.

4. Conclusiones

La tecnología de contenedores ha venido para quedarse, igual que ya lo hizo la virtualización.

Docker está orientado para fusiones de los equipos de Dev y Ops en DevOps, aportando mayor agilidad para las operaciones, la seguridad y mejorar los tiempos en los desarrollos, también permitir en una misma infraestructura el despliegue de aplicaciones con diferentes leguajes y filosofías, para que puedan interoperar entre ellas.

Esto permite aumentar la productividad de los equipos y reducir los tiempos de entrega

En el ámbito de las metodologías para el desarrollo, de la mano de los contenedores llegan las metodologías de integración continua y distribución continua. Esto promete mejorar los tiempos de desarrollo, ya que se acortan las esperas para pruebas y validaciones

Tendremos que realizar un análisis previo en el que se considere los equipos de desarrollo e infraestructura que tenemos, el uso que le vamos a dar a esta nueva plataforma, es decir, qué sistemas

de información se van a desplegar en la misma para elegir las diferentes opciones que tenemos. Queda reflejado, que, para entornos empresariales, siempre que sea posible se aconseja alguna de las soluciones comerciales frente a las soluciones de comunidad.

Habrá que estar pendiente de las nuevas actualizaciones de las plataformas, ya que constantemente están surgiendo nuevas herramientas para interoperar con los contenedores.

Agradecimientos

Me gustaría mostrar mi agradecimiento al profesorado y alumnado del Máster, que pese a las particularidades que durante este curso hemos sufrido por la Covid19 han conseguido llevar a buen puerto este Máster.

Una mención especial para mi familia, Amalia, Ángel y Mario, por acompañarme durante la realización del Master, y por aceptar con agrado mi falta de tiempo para ellos.

Referencias

[1] Tienda ReDIGIT Informática Circular, "Tecnología de virtualización basada en contenedores," *tienda ReDIGIT Informática Circular*, 2019.

[2] D. Russell, "https://cepymenews.es/predicciones-tecnologia-2020/," *cepymenews.es*, no. https://cepymenews.es/predicciones-tecnologia-2020/, 2020.

[3] E. Flo, "Teletrabajo, nube, contenedores y ciberseguridad protagonizarán 2021," *computerworld*, no. https://www.computerworld.es/tendencias/teletrabajo-nube-contenedores-y-ciberseguridad-protagonizaran-2021, 2021.

[4] Proyectos Wikimedia, "https://es.wikipedia.org/," 15 Diciembre 2020. [Online]. Available: https://es.wikipedia.org/wiki/Docker_(software).

[5] 2. L. a. d. Kubernetes, "https://kubernetes.io/," [Online]. Available: https://kubernetes.io/es/.

[6] Copyright © 2020 Red Hat, Inc, "https://www.openshift.com/," [Online]. Available: https://www.openshift.com/.

[7] A. E. Amri, "An Overall View On Docker Ecosystem — Containers, Moby, Swarm, Linuxkit, containerd & Kubernetes," 11 Junio 2018. [Online]. Available: https://medium.com/faun/an-overall-view-on-docker-ecosystem-containers-moby-swarm-linuxkit-containerd-kubernetes-5e4972a6a1e8.

[8] Erwin, "https://www.lomasnuevo.net/," 15 agosto 2016. [Online]. Available: https://www.lomasnuevo.net/cloud/maquinas-virtuales-vs-contenedores/.