Kurbenetes una mansión en la nube para tus aplicaciones

Yohan Carlos Camacho Santana, Fidelio Castillo Romero, Rosa Gómez Domínguez

Tecnológico Nacional de México / Instituto Tecnológico de Villahermosa, Tabasco

Resumen

El mundo tecnológico actual ha dado un gran salto desde que comenzó la Revolución Industrial, un momento de esplendor de grandes avances como la creación de la máquina de vapor en 1976. Existen numerosas tecnologías y avances que han surgido desde aquellos años hasta nuestros días. Uno de los mayores avances del momento son las computadoras y el software que los complementan. El software es algo que nos rodea, está presente en cualquier parte de la vida moderna, desde la industria hasta el hogar. Siempre ha existido residente en un dispositivo, ya sea un teléfono celular, un televisor inteligente o un reloj de pulsera. No fue sino hasta hace unos diez años que surgió la idea de buscar una manera de que el software exista en varias computadoras. Piezas de códigos divididos en muchas partes pero que mantienen comunicación entre sí, trabajando, existiendo y creciendo siempre que queramos como si no hubiera límites. Software al que el usuario final puede acceder y usar en todo momento, independientemente de cuándo o dónde se encuentre, siempre que tenga una conexión a la red de Internet.

Abstract

The current technological world has taken a great leap since the Industrial Revolution began, a moment of splendor of great advances such as the creation of the steam engine in 1976. There are numerous technologies and advances that have emerged from those years to the present day. One of the greatest advances of the moment are the computers and the software that complement them. Software is something that surrounds us, it is present in any part of modern life, from industry to the home. There has always been a resident of a device, be it a cell phone, a smart TV or a wristwatch. It wasn't until about ten years ago that the idea came up to find a way for software to exist on multiple computers. Pieces of codes divided into many parts but that maintain communication with each other, working, existing and growing whenever we want as if there were no limits. Software that the end user can access and use at all times, regardless of when or where they are, as long as they have a connection to the Internet.

Palabras claves: *Kubernetes, Dockers, contenedores, sistemas distribuidos, microservicios.*
Keywords: Kubernetes, Dockers, containers, distributed systems, microservice.

1. INTRODUCCIÓN

A partir del siglo XX el desarrollo vertiginoso de las tecnologías ha sumergido al mundo en una era de miniaturización tecnológica. Con la sustitución de los bulbos al vacío por el transistor, posteriormente la invención del circuito integrado y la creación del microchip se comienza a trabajar camino a hacia esta era. Dichos componentes dieron como consecuencia que las computadoras y los equipos de cómputo se fueran haciendo pequeños, hasta el punto de poder llevarse en un bolsillo, disminuyendo en grosor, tamaño y peso. Aparecieron gran cantidad de equipos electrónicos y computadoras portátiles que hacen la vida más fácil, debido a que el hombre trataba de simplificar su vida ahorrando tiempo y esfuerzo, pudiendo así incorporar una serie de tareas en un solo dispositivo.

Con el surgimiento de Internet, se comenzaron a crear tecnologías compatibles con dichos medios, el mundo de las comunicaciones cambió drásticamente naciendo nuevas formas de intercambio verbal, manifestándose

en forma de correo electrónico, redes sociales, chats y video-conferencias. Se renovó el comercio haciéndolo más amplio y dinámico, creándose el comercio electrónico el cual fomentó los pagos mediante dispositivos y compras a distancia. Toda esta renovación de software y hardware ha desembocado en inconvenientes de mantenimiento, dificultades cuando se requiere crecer en capacidad y obstáculos si se desea cambiar de un lugar a otro el software ya desplegado. Debido a que se requieren librerías u otras instalaciones que en ocasiones no se encuentran disponibles en el lugar al cual migraremos el software. En cuanto a la arquitectura las utilizadas por excelencia hasta hace unos años era la monolítica y cliente-servidor. La primera por su parte define que la aplicación reside en un solo lugar, en este caso un computador. Este tipo de construcciones de software hacen que sea extremadamente lento realizar cambios en la aplicación además de no poder efectuar modificaciones de forma transparente para el usuario que las utiliza. Por otra parte, es posible que se requiera una nueva versión de software completamente reconstruida cada vez que se realicen pequeños cambios en las funcionalidades del sistema.

La arquitectura cliente-servidor llegó para resolver algunos de los problemas que posee la monolítica, pero aún sigue siendo complicado en cuanto a migraciones y mantenimiento a medida que el software crece. En este tipo de estructura existe un software en una computadora generalmente un servidor y otro software en el dispositivo del usuario (Cliente), el funcionamiento es guiado por peticiones realizadas desde el cliente hacia servidor y las respuestas dadas por este último hacia el origen de las consultas en este caso el cliente. Este tipo de arquitectura es posible con la existencia de las redes computacionales e internet, pero ¿Qué pasaría si el servidor se estropea y deja de funcionar?, por supuesto que dejaría inhabilitado el sistema y se interrumpiría todas las peticiones hacia el mismo. Entonces sería de gran ayuda un tipo de arquitectura que la aplicación pudiera existir en muchas computadoras a la vez funcionando como un todo, igualmente que pudiera sobrevivir a que unas o varias de estas máquinas que integran el todo se pudieran reponer en caso de problemas y el sistema no dejara de funcionar. Si existiera la necesidad de implementar nuevas funcionalidades al sistema o realizar mantenimientos a las ya existentes fuera de forma transparente para el usuario final, es decir que el usuario ni se daría cuenta de que se está migrando las bases de datos de la aplicación hacia un servidor en Europa, por ejemplo. Todas estas posibilidades son brindadas actualmente por un tipo de arquitectura que se conoce ya hace varios años, pero gracias al desarrollo de las tecnologías actuales se ha podido implementar más eficientemente que antaño.

Esta forma de interconectar ordenadores implementa la filosofía de divide y vencerás, es mucho mejor tener sistemas divididos en pequeñas porciones de códigos que se comunican e interactúan entre si funcionando al final como un único sistema. Al mismo tiempo de no estar residente en una sola computadora, sino que está distribuido en muchas de ellas. Este tipo de arquitecturas trajo como resultado lo que se conoce hoy en día como sistemas distribuidos y aplicaciones basadas en microservicios.

2. DESARROLLO

¿Qué es un microservicio y para que lo necesitamos?

Según la empresa de software libre RedHat "Los microservicios representan un estilo de arquitectura y un modo de programar software. Con los microservicios, las aplicaciones se dividen en sus componentes más pequeños, y son independientes entre sí". (RedHat, 2020). O bien como la empresa Microsoft precisa que "Los microservicios son pequeños e independientes, y están acoplados de forma imprecisa. Un único equipo reducido de programadores puede escribir y mantener un servicio… Los servicios pueden implementarse de

ISSN: 2007-4786

manera independiente. Esto difiere del modelo tradicional, donde una capa de datos independiente controla la persistencia de los datos" (Wasson, 2019).

Como se evidenció los microservicios son independientes y a su vez se relacionan con otros semejantes mediante el intercambio de mensajes. Los mismos no están puramente habilitados en los sistemas distribuidos, sino que son encapsulados dentro de unidades de software. Las unidades son capaces de empaquetar todo el código del microservicio y las librerías de software necesarias para que este funcione. Posibilitando la ejecución de manera veloz y eficiente en cualquier otro sistema donde se deseen migrar. A dichas unidades de software se les llamó contenedores. Esta idea revolucionaria de crear los contenedores multiplataforma fue desarrollada por Solomon Hykes, fundador de Docker. (Solomon Hykes, 2019)

Una imagen de contenedor Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, herramientas del sistema, bibliotecas del sistema y configuraciones. Las imágenes se convierten en contenedores cuando se ejecutan en Docker Engine motor de software, el cual es el encargado de administrar los mismos.

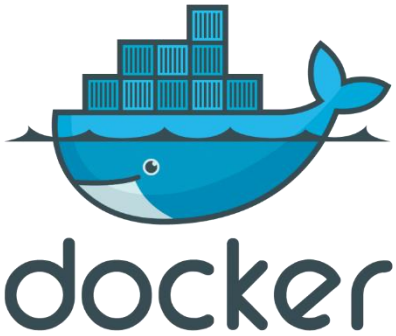

Ilustración 1. *Logotipo del proyecto Docker.* *(Shadow, 2015)*

El software en contenedores siempre se ejecutará igual, independientemente de la infraestructura o el sistema operativo utilizado para proporcionar la operatividad de la aplicación. Los contenedores aíslan el software de su entorno y aseguran que funcione de manera uniforme a pesar de las diferencias, por ejemplo, entre desarrollo y puesta en escena, además de poseer la capacidad de poder desplegarse de forma local o para lo que fueron creados en un principio, en la nube. (Docker Inc, 2020)

Los contenedores no son nuevos, pero su uso para implementar aplicaciones fácilmente sí lo es. Algunas de las ventajas que proporciona este tipo de desarrollo son:

- Flexible: incluso las aplicaciones más complejas se pueden contener en contenedores.
- Ligero: los contenedores aprovechan y comparten el núcleo del host, lo que los hace mucho más eficientes en términos de recursos del sistema que las máquinas virtuales.
- Portátil: puede construir localmente, implementar en la nube y ejecutar en cualquier lugar.
- Bajo acoplamiento: los contenedores son altamente autosuficientes y encapsulados, lo que le permite reemplazar o actualizar uno sin interrumpir otros.
- Escalable: puede aumentar y distribuir automáticamente réplicas de contenedor en un centro de datos.
- Seguro: los contenedores aplican restricciones agresivas y aislamientos a los procesos sin ninguna configuración requerida por parte del usuario. (McKee, 2019)

ISSN: 2007-4786

A medida que la aplicación crece también aumenta en número la cantidad de contenedores necesarios para implementar, gestionar y administrar, estas tareas suelen ser muy tediosas, por lo que se echaba de menos algún software que pudiera encargarse de orquestar todos estos instrumentos, metafóricamente hablando para lograr una sincronización entre ellos y poder ejecutar la melodía para la que fueron creados. El sistema Borg fue el pionero en la gestión de contenedores de código abierto creado por la empresa Google, era un sistema de administración capaz de gestionar disímiles servidores a la vez (Clusters) y ejecutar miles de tareas. Borg fue sucedido luego por Omega el cual era mucho más eficiente y veloz. Luego de procesos de mejoras continuas se anhelaba dotar al proyecto con una interfaz de usuario elegante, simple y fácil de usar. Solo hicieron falta tres meses para crear un prototipo listo para ser compartido y probado. Kubernetes como fue bautizado el proyecto final, ahora está implementado en miles de organizaciones, con el apoyo de más de 830 contribuyentes. (Craig McLuckie, 2016)

Kubernetes es capaz de implementar y administrar fácilmente aplicaciones en contenedores. Está basado en las características de estos para ejecutar aplicaciones heterogéneas sin tener que conocer ningún detalle interno de las mismas. Además de carecer de implementación manual de estas aplicaciones en cada computador en lo adelante host. Debido a que las aplicaciones se ejecutan en contenedores, no producen afectaciones sobre ninguna otra aplicación que se encuentre en ejecución en el mismo servidor.

Kubernetes le permite ejecutar sus aplicaciones de software en miles de nodos de computadora como si todos esos nodos fueran una sola computadora enorme. Se abstrae la infraestructura subyacente simplificando así el proceso, la implementación y gestión tanto para el desarrollo de las aplicaciones como para los equipos de operaciones. El sistema está compuesto por un nodo maestro y cualquier número de nodos de trabajos. El nodo maestro aloja el plano de control del sistema, brinda la capacidad de controlar y gestionar toda la orquestación del mismo.

Los nodos de trabajo son los encargados de ejecutar los contenedores de aplicaciones que les son asignados por el máster. Cuando el desarrollador envía una lista de aplicaciones al maestro, Kubernetes las implementa en el grupo de trabajadores nodos. Esto se lleva a cabo sin importar en qué nodo o grupos de nodos recaiga la implementación. El desarrollador puede especificar cuáles aplicaciones deben ejecutarse juntas y el sistema las desplegará en el mismo nodo de trabajo. Por otra parte, lo que quede de la aplicación se extenderán alrededor de otros grupos de nodos, pero siguiendo una estrecha comunicación entre ellos, independientemente de dónde estén desplegados.

ISSN: 2007-4786

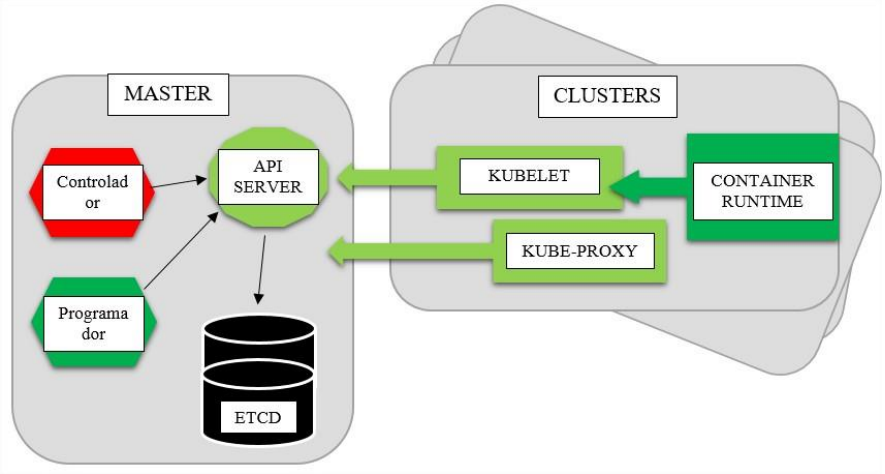

Ilustración 2. Interacción entre componentes que integran Kubernet.

El uso de Kubernetes para administrar las aplicaciones implementadas también significa que no se necesita monitorear constantemente la carga de aplicaciones individuales para reaccionar a picos de carga repentinos. Si Kubernetes se ejecuta en la infraestructura de la nube, donde agregar nodos adicionales es tan fácil como solicitarlos a través de la API del proveedor de la nube, Kubernetes puede incluso escalar automáticamente todo el tamaño del clúster hacia arriba o hacia abajo según las necesidades de las aplicaciones desplegadas.

Si se recurre al hecho de que las aplicaciones se ejecutan en el mismo entorno tanto en el desarrollo como la producción, esto tiene un gran efecto cuándo se descubren errores, pudiendo así ser corregidos una sola vez. Cuanto antes sea descubierto un error en la aplicación, más fácil será solucionarlo y su reparación requerirá de mucho menos trabajo.

Los desarrolladores son los encargados de subsanar todos los problemas en cuanto al software que se encuentra en ejecución, por lo que significa mucho menos trabajo y más velocidad al corregir estos detalles de último momento. Por último, también se debe considerar el aumento de la confianza que los desarrolladores sentirán sabiendo que cuando una nueva versión de su aplicación fuese implementada, Kubernetes puede detectar automáticamente si el nuevo software presenta problemas de compatibilidad con las partes ya existentes y detiene su lanzamiento de inmediato.

3.- CONCLUSIONES

Estas tecnologías en crecimiento y pleno esplendor conllevarán quizás a una nueva época de avances, miles de aplicaciones son desplegadas todos los días en disímiles centros de datos que en ocasiones se encuentran interconectados con otros por todo el mundo. Se infiere que sería posible realizar softwares capaces de vivir en todas partes, a veces podemos encontrar paranoias como las de espionaje y que todos somos observados.

En un mundo cada vez más globalizado e interconectado cualquier suceso es posible haciendo un uso indebido de las tecnologías, por eso resulta de vital importancia hacer conciencia en la ética tecnológica y como destinar estos avances al bien común, desechando las malas prácticas que a veces algunos cometen. Pero viendo el lado bueno de todo, pudieran existir softwares inteligentes que previeran catástrofes por todo el mundo, capaces de prevenir sunamis, terremotos, volcanes entre otros.

Carreteras inteligentes que puedan cargar tu auto eléctrico a medida que avanza y reportar averías si se estropea algo o en caso de accidente. Identificaciones médicas universales que sin importar donde estés, el personal de salud pueda conocer enfermedades crónicas, tipo de sangre o diferentes apoplejías en caso de algún accidente deje incapacitado a la persona. Un sin número de aplicaciones pueden ser conferidas de inteligencia que pueda ayudar en la vida cotidiana y las ciencias.

Todo este despliegue de tecnologías e inventivas dotadas de grandes avances realzan una vez más que el ingenio humano no tiene límites. De requerirse ampliar aún más los conocimientos en el tema se puede consultar el libro Kubernetes: Up and Running de Helsey Hightower, Brendan Burns y Joe Beda un buen referente para comenzar a implementar este tipo de tecnología.

REFERENCIAS

[1] Craig McLuckie. (2016, julio 22). *From Google to the world: The Kubernetes origin story*. Google Cloud Blog. https://cloud.google.com/blog/products/gcp/from-google-to-the-world-the-kubernetes-origin-story/

[2] Docker Inc. (2020). *What is a Container? | App Containerization | Docker*. What Is a Container? https://www.docker.com/resources/what-container

[3] McKee, P. (2019, diciembre 3). *Orientation and setup*. Docker Documentation. https://docs.docker.com/get-started/

[4] RedHat. (2020). *¿Qué son los microservicios?* Ventajas de la arquitectura de microservicios. https://www.redhat.com/es/topics/microservices

[5] Shadow, A. (2015, mayo 9). Docker—A highly portable and lightweight software container. *Shadowandy - My Life Stories*. https://www.shadowandy.net/2015/05/docker-a-highly-portable-and-lightweight-software-container.htm

[6] Solomon Hykes: The Founder of Enterprise Container Platform, «Docker». (2019, septiembre 11). *Your Tech Story*. https://www.yourtechstory.com/2019/09/11/solomon-hykes-founder-enterprise-container-platform-docker/

[7] Wasson, M. (2019, octubre 30). *Estilo de arquitectura de microservicios—Azure Application Architecture Guide*. https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

Correo electrónico autor: *yccamacho87@aol.com*