Arquitectura Cloud Native

Juan Manuel Castillo Nievas

1. ¿Qué es Cloud Native?

La arquitectura Cloud Native es un patrón de arquitectura software usado para el desarrollo de aplicaciones usando los principios del Cloud Computing. Nace con el objetivo principal de hacer desarrollos de aplicaciones mucho más rápidos y de mayor calidad a un precio menor [3]. Esto se debe a que se describen entornos que se encuentran en contenedores que ayudan a tener aplicaciones basadas en microservicios dentro de contenedores orquestados por kubernetes [4].

El enfoque de esta arquitectura está centrada en el desarrollo de microservicios que sean independientes y que no se ejecuten en local, sino que se usen plataformas dinámicas basadas en contenedores. Pero Cloud Native no sólo significa construir microservicios en Docker, si no que se disponen de metodologías, herramientas, estándares, estilos de arquitectura e infraestructura, e independencia de los proveedores de cloud [6].

Hay varias definiciones de Cloud Native, pero una de ellas es la siguiente [4]: *"El término de cloud native designa un modelo de desarrollo de software según el cual las aplicaciones se diseñan para ejecutarse en la nube desde el principio. El resultado son aplicaciones nativas en la nube o NCA (del inglés native cloud application), capaces de aprovechar al máximo todos los puntos fuertes de la arquitectura de la computación en la nube."*

2. Beneficios de Cloud Native

Los beneficios principales de este nuevo paradigma son los siguientes [1] [7]:

- Escalabilidad: escalar las aplicaciones tanto horizontal como verticalmente de una manera mucho más rápida y dinámica, teniendo muy pocas restricciones en lo que se refiere al hardware.

- Velocidad: esta arquitectura es capaz de aumentar la capacidad de los equipos usados sin aumentar los recursos que usan. También se hace un desarrollo, testeo y entrega de código de una manera mucho más rápida.

- Costos marginales: se aprovecha la facturación flexible de los proveedores cloud para pagar los recursos adicionales sólamente cuando sea necesario.

- Conexión de datos: se ofrece la posibilidad de crear grupos de datos conectados haciendo que se pueda hacer un análisis más profundo y sofisticado, mejorando de esta forma la colaboración usando nuevas plataformas y herramientas y permitiendo

tomar decisiones de negocio de una manera más rápida.

- Ecosistemas DevOps: el término DevOps se describe más adelante, pero a modo de introducción se puede decir que en términos de Cloud Native, esta nueva metodología ofrece una mayor agilidad, conectividad y transparencia en los procesos de las entidades.

- Mejora de la experiencia del cliente: Cloud Native ha introducido una nueva cultura Data Centric que permite a las compañías disponer o replicar sus datos con una velocidad de reacción muy grande. De esta manera, los clientes finales no sufren ningún tipo de demora y, por lo tanto, mejorando su experiencia final. Las empresas pueden gestionar las interacciones puntuales y orquestar todo el Customer Journey (mapa del recorrido del cliente).

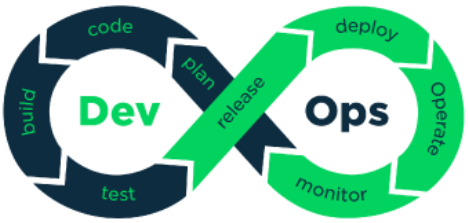
- Filosofía "*As a Service*": esto es algo que ya está presente en Cloud Computing y de lo que ya sabemos todos los beneficios que trae, pero en Cloud Native estos beneficios se incrementan aún más, ya que son aplicaciones que tienen una capacidad de adaptación enorme a todos los cambios que haya en el mercado. "As a Service" es sinónimo de innovación en tiempo real.

3. Los 4 pilares de Cloud Native

Este patrón de arquitectura hace uso de las ventajas que ofrecen los 4 pilares de los que dispone: DevOps, entrega continua, microservicios y contenedores.

3.1 DevOps

Esta nueva metodología nace de la necesidad de tener una visión compartida del producto, que al fin y al cabo es lo más importante [1]. Años atrás, el equipo de desarrollo y de operaciones trabajan de forma separada. A esto se le suma que ambos equipos pueden tener objetivos muy diferentes entre ellos, y trabajar de forma aislada sólo complica más todo el proceso del producto.

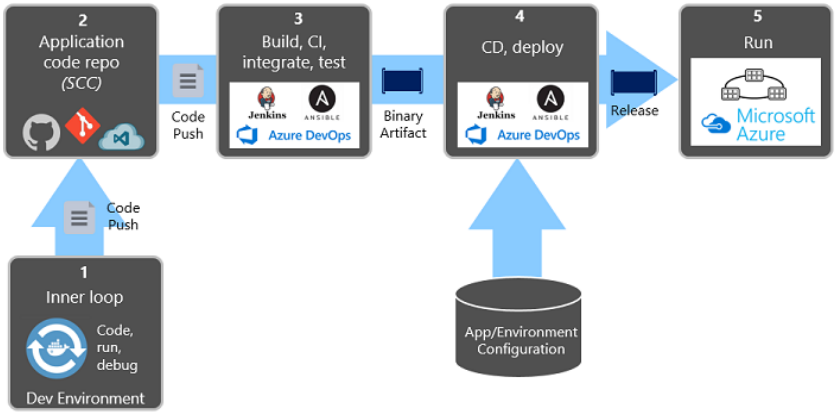


Esta metodología combina ambos equipos: Dev (desarrollo de software) y Ops (operaciones). De esta manera, se comparte la misma visión y los mismos objetivos haciendo que se mejoren y maximicen las comunicaciones entre ambos equipos para producir y probar el software de una forma mucho más eficiente [2]. Esto permite entregar el software de forma más rápida, con una mayor calidad y un menor coste, aumentando la frecuencia de los lanzamientos o *releases*.

3.2 Continuous Delivery (Entrega continua)

Esto implica la automatización de las tareas de construcción y despliegue. En la publicación de Miguel Garrido [1] cuenta su experiencia trabajando a principios de los años 2000. Tenía que apuntar en una libreta cada fichero que tocaba para desarrollar una nueva rama del producto y posteriormente identificar cuáles debía subir a mano para luego solicitar por correo al departamento de operaciones de la empresa correspondiente que los pusiera en el sitio correspondiente.

Está claro que esta metodología ya no sirve hoy en día. La entrega continua permite a los equipos de desarrollo automatizar todo el proceso de despliegue, mejorando de esta forma los tiempos de testeo, costos, escalabilidad y despliegue de código en cualquier fase del ciclo de desarrollo [2]. Esta automatización también implica que habrá menos errores humanos.



3.3 Microservicios

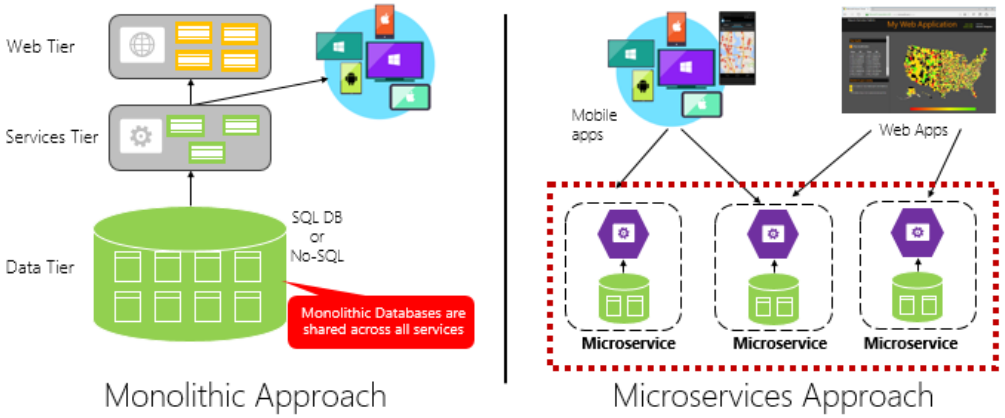
Los microservicios son un enfoque de arquitectura Cloud Native en el que una aplicación está compuesta de varios servicios más pequeños que son independientes entre sí. Los microservicios ofrecen las siguientes ventajas [1]:

- Gestión eficiente de recursos: el escalamiento se hace de forma automática, ajustando el consumo en memoria y CPU dependiendo de la demanda a nuestros microservicios.

- Desacoplamiento entre funcionalidades: como todos sabemos, cada microservicio debe cubrir una única necesidad, de esta manera se evita el tener un mismo fichero con miles líneas de código realizando distintas funciones de una manera mucho más desorganizada.

- Equipos DevOps independientes y reducidos: debido a que existe el desacoplamiento entre funcionalidades, implementar un cambio en una funcionalidad no implica propagar ese error a otra funcionalidad, puesto que ahora son independientes. Esto permite que los equipos DevOps realicen de una manera más eficiente la entrega

continua y se garantice una mayor calidad del producto.

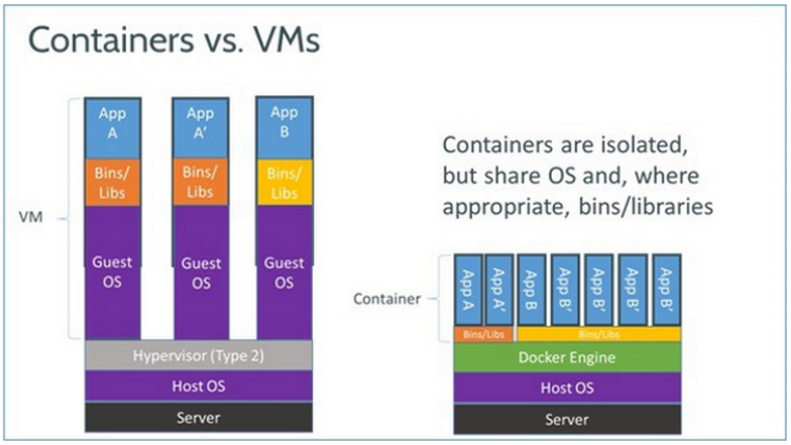


3.4 Contenedores

Tal y como se ha visto en clase a lo largo del cuatrimestre, los contenedores garantizan que el producto se puede ejecutar de la misma manera en cualquier entorno que se despliegue. De esta manera, la pregunta "¿por qué en mi ordenador no funciona?" al ejecutar el producto en entornos productivos o de test desaparece por completo [1].

De forma muy resumida, los contenedores son como un sistema operativo que contiene lo justo y necesario para la ejecución de una aplicación. El código que se debe ejecutar está empaquetado con todas las librerías y dependencias necesarias que se necesitan, por lo tanto se ahorra muchísimo espacio al no tener otras librerías que en ningún momento se llegan a utilizar.

A diferencia de las máquinas virtuales, los contenedores comparten capas del mismo sistema operativo, por lo tanto esto hace que los contenedores sean mucho más ligeros. Es muy similar a una máquina virtual muy portable y menos exigente en cuanto a recursos de cómputo [3].



4. Casos de uso

Las empresas están evolucionando y cada día son más las empresas que implementan esta arquitectura, ya que les permite responder de manera muy rápida a las nuevas condiciones del mercado. Algunas de las empresas muy conocidas que implementan esto son [5]:

- Netflix: tiene más de 600 servicios en producción e implementa cientos de veces al día.

- Uber: tiene más de 1.000 servicios en producción e implementa varios miles de veces cada semana.

- WeChat: tiene más de 3.000 servicios en producción e implementa 1.000 veces al día.

Cabe mencionar también la empresa de Trainingym de la que tuvimos una charla en clase. Nos contaron cómo nacieron en 2011 cuando aún todo esto de Cloud Native aún no estaba pensado y cómo han tenido que "modernizarse" debido a los constantes cambios en el mercado y a todas las ventajas que ofrecen, tanto en su desarrollo como en la producción final de cara al cliente.

Bibliografía

[1] https://www.paradigmadigital.com/techbiz/haz-despegar-tu-negocio-desarrollando-aplicaciones-cloud-native/
[2] https://www.ibm.com/co-es/cloud/cloud-native
[3] https://www.aplyca.com/es/blog/cloud-native
[4] https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-cloud-native/
[5] https://docs.microsoft.com/es-es/dotnet/architecture/cloud-native/definition
[6] https://medium.com/ingenia-architectural-journeys/que-entendemos-por-cloud-native-d781e569c776
[7] https://atmira.com/5-razones-modernizacion-cloud-native/