ARQUITECTURA MODERNA DE APLICACIONES EN LA NUBE

C. A. Ramírez García
7690-21-10603 Universidad Mariano Gálvez
Seminario de Tecnologías de Información
cramirezg19@miumg.edu.gt

Repositorio

https://github.com/CarlosRamirez2003/FORO-ACADEMICO-4.git

Resumen

En los últimos años el desarrollo de software ha cambiado de forma acelerada debido a la necesidad de crear aplicaciones más rápidas, seguras y fáciles de mantener. Las empresas y los usuarios esperan que los sistemas funcionen en todo momento, que puedan crecer cuando la demanda aumenta y que al mismo tiempo se proteja la información. Para responder a estas exigencias han surgido distintas prácticas y herramientas que buscan hacer más sencilla la vida de los equipos de tecnología. La automatización de procesos en servidores, la administración de contenedores, la división de las aplicaciones en componentes pequeños, los mecanismos de acceso seguro y los lineamientos para trabajar en la nube son algunos ejemplos de este cambio. Todas estas propuestas tienen en común que ayudan a reducir errores, facilitan el trabajo en equipo y preparan los sistemas para enfrentar entornos cada vez más complejos. Aunque aprender a usarlas puede ser un reto, representan un paso importante hacia aplicaciones más confiables, escalables y con mayor capacidad de adaptación a los cambios que exige el mundo digital actual.

Palabras clave: orquestación, kubernetes, microservicios, oauth2, 12-factor

Orquestación de servidores

Antes, los administradores de sistemas tenían que entrar manualmente a cada servidor para instalar programas, modificar configuraciones o reiniciar servicios. Este método funcionaba cuando se tenía una o dos máquinas, pero en la actualidad los sistemas funcionan sobre decenas o incluso cientos de servidores, y hacerlo a mano se vuelve imposible. Además, la probabilidad de cometer un error aumenta mucho, y un pequeño fallo puede causar que todo el sistema quede fuera de servicio.

La orquestación de servidores surge para resolver este problema. Con ella, las tareas repetitivas se automatizan, lo que significa que en lugar de entrar servidor por servidor, se define un conjunto de instrucciones y las herramientas se encargan de aplicarlas de forma uniforme en todos los equipos. Por ejemplo, si se necesita instalar una nueva versión de una aplicación en cien servidores, la orquestación lo hace en minutos y con la seguridad de que todos quedarán configurados igual.

Existen varias herramientas que permiten este trabajo. Ansible, por ejemplo, utiliza archivos de texto en formato YAML, fáciles de leer, donde se describen las tareas a realizar. Puppet trabaja con un enfoque declarativo, en el que se define el estado que se desea y el sistema se encarga de alcanzarlo. Chef usa recetas escritas en Ruby para especificar cómo deben configurarse los servidores. Todas ellas tienen el mismo objetivo: reducir el trabajo manual, mejorar la consistencia y ahorrar tiempo.

Lo más importante es que la orquestación no solo hace más eficiente el trabajo, también ayuda en seguridad y confiabilidad. Al tener configuraciones estandarizadas, se evita que cada servidor esté "personalizado" y se reduce el riesgo de errores humanos.

Kubernetes como estándar de contenedores

Los contenedores cambiaron la manera de desplegar software porque permiten ejecutar aplicaciones en entornos aislados, ligeros y fáciles de mover de un lugar a otro. Sin embargo, cuando se empieza a trabajar con docenas o cientos de contenedores, surge el problema de cómo organizarlos, actualizarlos y asegurarse de que siempre estén funcionando.

Aquí entra Kubernetes, que se ha convertido en el estándar mundial para manejar contenedores. Esta plataforma administra de manera automática los recursos de los servidores y decide dónde debe ejecutarse cada contenedor. Si uno falla, Kubernetes lo reinicia. Si hay demasiada carga, puede arrancar más copias para repartir el trabajo. Además, permite actualizar aplicaciones sin interrumpir el servicio, lo cual es vital para empresas que no pueden darse el lujo de quedar fuera de línea.

Kubernetes se basa en varios conceptos. Los pods son la unidad mínima de ejecución, que puede contener uno o varios contenedores. Los deployments facilitan actualizaciones ordenadas y seguras. Los services permiten que los contenedores se comuniquen entre sí y con el exterior de manera estable. Y los namespaces sirven para organizar y separar recursos dentro de un mismo clúster.

Aunque al principio puede parecer complicado, Kubernetes es muy flexible. Se adapta tanto a empresas pequeñas que apenas comienzan a usar contenedores como a grandes compañías con miles de servicios en la nube. Es por eso que hoy se ha convertido en una de las tecnologías más demandadas en el mundo de TI.

Microservicios y escalabilidad empresarial

Durante muchos años, las aplicaciones se construían de forma monolítica: todo el sistema en un solo bloque de código. Esto hacía que fueran más simples de desarrollar al inicio, pero muy difíciles de escalar y mantener con el paso del tiempo. Si una parte del sistema fallaba, era común que todo quedara inactivo.

La arquitectura de microservicios propone una forma distinta: dividir la aplicación en servicios pequeños, independientes y especializados. Por ejemplo, en una tienda en línea podría existir un microservicio de pagos, otro de inventario, otro de usuarios y otro de envíos. Cada uno puede desarrollarse, probarse y desplegarse por separado. Si hay más demanda en los pagos, se pueden duplicar solo esos servicios sin necesidad de aumentar los demás.

Una de las grandes ventajas es la independencia tecnológica: cada microservicio puede escribirse en el lenguaje o framework más conveniente. Esto facilita la innovación y permite que distintos equipos trabajen en paralelo. Además, si un servicio falla, los demás siguen funcionando, lo que mejora la resiliencia.

Sin embargo, los microservicios también traen nuevos retos. Al haber tantos componentes, la comunicación entre ellos se vuelve compleja y requiere soluciones como mensajería asíncrona o APIs bien diseñadas. También es más difícil llevar un control centralizado de los datos, ya que cada servicio puede tener su propia base de datos.

Aun con estas dificultades, los microservicios son hoy la arquitectura favorita de muchas empresas grandes, como Netflix, Amazon o Uber, que necesitan sistemas capaces de crecer sin parar.

OAuth 2.0 y seguridad en aplicaciones distribuidas

En internet es muy arriesgado que los usuarios compartan sus contraseñas con cada aplicación que desean usar. Para resolver este problema nació OAuth 2.0, un protocolo que permite otorgar acceso limitado a los recursos de un usuario sin necesidad de compartir su clave.

El funcionamiento se basa en tokens temporales. En lugar de que la aplicación tenga la contraseña, obtiene un permiso especial emitido por un servidor de autorización. Así, si el token se filtra, solo da acceso por un tiempo o a un recurso específico, y no compromete toda la cuenta.

Un ejemplo común es cuando una aplicación pide iniciar sesión con Google o Facebook. El usuario no le da la contraseña a la aplicación, sino que autoriza a través de Google, y esta entrega un token con permisos limitados. Esto protege mejor los datos y hace más segura la integración entre distintos servicios.

OAuth 2.0 es especialmente importante en arquitecturas modernas donde los microservicios necesitan comunicarse entre sí y con terceros. Permite un control más fino de quién accede a qué información, algo esencial en la era actual donde la seguridad es tan crítica.

Observaciones y comentarios

La adopción de estas tecnologías refleja la evolución natural del software hacia sistemas más ágiles y distribuidos. Sin embargo, es evidente que su implementación exige preparación técnica y cambios culturales en las organizaciones.

Las empresas deben invertir no solo en infraestructura, sino también en capacitación constante de su personal, ya que sin el conocimiento adecuado las herramientas pierden efectividad.

Conclusiones

Considero que la evolución hacia arquitecturas distribuidas y automatizadas es una necesidad más que una opción para las organizaciones modernas.

Me parece que el verdadero reto no está en la tecnología, sino en la capacidad de los equipos para adaptarse y aprender a trabajar con estas nuevas herramientas.

Opino que la combinación de buenas prácticas y una mentalidad de mejora continua es lo que realmente marca la diferencia en el éxito de los proyectos tecnológicos.

E-Grafia

Kubernetes. (s. f.). Documentación oficial de Kubernetes en español. Kubernetes.io. Recuperado de https://kubernetes.io/es/docs/home/

Red Hat. (2023). Documentación de Ansible Automation Platform. Red Hat, Inc. Recuperado de https://www.redhat.com/es/technologies/management/ansible

Wiggins, A. (2011). La aplicación de los doce factores. Heroku. Recuperado de https://12factor.net/es/

Red Hat. (s. f.). ¿Qué son los microservicios?. Red Hat, Inc. Recuperado de https://www.redhat.com/es/topics/micro
are-microservices