CLOUD NATIVE

J. P. Araiz López
7690-17-9477 Universidad Mariano Gálvez
Seminario de Tecnologías de Información
jaraizl@miumg.edu.gt

23 de agosto de 2024

Resumen

En la actualidad se están utilizando entornos orientados a la nube, lo cual ha elevado la importancia de la gestión de servidores, los enfoques arquitectónicos orientados a microservicios y el uso de contendores, por lo que conocer sobre herramientas que puedan ayudar a la gestión de estos como lo es Kubernetes es importante como también los es conocer metodologías enfocadas al desarrollo en la nube como el 12-Factor APP que nos proporciona una guía de buenas prácticas para estos desarrollos modernos orientados a los microservicios, contendores y servicios en la nube. Por lo que en este artículo conoceremos un poco más sobre estos temas.

Palabras Clave

Servicios, Automatización, Contenedores

Orquestación de servidores

La orquestación de servidores hace referencia a la coordinación y gestión automatizada de diferentes servidores para garantizar que funcionan de manera eficiente, a modo de realizar una serie de tareas o procesos de forma ordenada. La orquestación de servidores sirve para simplificar y agilizar las operaciones informáticas, permitiendo gestionar y escalar infraestructuras de forma eficiente.

La orquestación de servidores se basa en herramientas y plataformas de gestión que permiten definir y ejecutar flujos de trabajo automatizados. Por lo general la orquestación se realiza a través de un controlador centralizado que coordina y ejecuta todos los flujos definidos, comunicándose con los servidores y recursos involucrados para realizar las tareas de forma automatizada.

La orquestación de servidores ofrece los siguientes beneficios:

- Eficiencia operativa
- Escalabilidad y flexibilidad
- Consistencia y conformidad

- Agilidad y rápida implementación

Kubernetes

Para conocer de Kubernetes antes debemos hablar de los contenedores, estos son componentes de aplicaciones livianos y ejecutables que combinan el código fuente de las bibliotecas y dependencias de un sistema operativo que son necesarias para ejecutar el código en cualquier entorno.
Kubernetes es una plataforma portable de código abierto para la administración de cargas de trabajo y servicio, que facilita la automatización y la configuración declarativa. Tiene un ecosistema grande y de rápido crecimiento. Podemos decir que Kubernetes programa y automatiza tareas realacionadas con contenedores a lo largo del ciclo de vida de la aplicación.
Kubernetes no es una plataforma como servicio (PaaS) convenicional, ya que opoera a nivel de contenedor y no a nivel de hardware. Kubernetes no es monolítico y las soluciones que se ofrecen de forma predeterminada son opcionales e intercambiables.

Kubernetes se utiliza para crear aplicaciones fáciles de administrar e implementar en cualquier lugar. Cuando está disponible como un servicio administrado te ofrece una gran variedad de soluciones según sean tus necesidades.

Los beneficios que ofrece Kubernetes serían:

- Operaciones automatizadas: cuenta con comandos para manejar gran parte del trabajo pesado que forma parte de la administración de aplicaciones, lo que permite automatizar gran parte de las operaciones diarias.
- Abstracción de la infraestructura: se encarga del procesamiento, las herramientas de redes y el almacenamiento en nombre de tus cargas de trabajo.
- Supervisión del estado de los servicios: ejecuta verificaciones de estado de manera continua de todos los servicios.

Microservicios

Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software, donde este se compone por pequeños servicios independientes que se comunican entre ellos mediante API bien definidas. Esta arquitectura hace que las aplicaciones sean más fáciles de escalar y más rápidas de desarrollar, permitiendo la innovación y la aceleración del tiempo de comercialización de las nuevas características. Actualmente los microservicios están muy ligados al concepto de contenedor, una unidad de software que empaqueta el código junto a todas sus dependencias para que el servicio se ejecute de forma rápida en cualquier entorno informatico.

Los microservicios poseen las siguientes características:

- Autónomos: cada servicio o componente se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otro servicio.

- Especializados: cada servicio o componente está diseñado o enfocado a resolver un problema en específico.

Los beneficios de los microservicios:

- Agilidad
- Escalado flexible
- Implementación sencilla
- Libertad tecnológica
- Código reutilizable
- Resistencia

OAuth2.0

Oauth que significa "Open Authorization", es un estándar diseñado para permitir que un sitio o aplicación acceda a recursos alojados por otras aplicaciones en nombre de su usuario. Este es un protocolo de autorización y no un protocolo de autenticación, como tal está diseñado principalmente como un medio para conceder acceso a un conjunto de recursos utilizando para ello tokens de acceso. Un token de acceso es un dato que representa la autorización para acceder a estos recursos en nombre del usuario final.

OAuth2.0 cuenta con roles que definen los componentes esenciales de un sistema de OAuth 2.0 y son los siguientes:

- Propietario del recurso: El usuario o sistema que posee los recursos protegidos y puede conceder acceso a ellos.
- Cliente: El cliente es el sistema que requiere acceso a los recursos protegidos. Para acceder a los recursos, el cliente debe poseer el token de acceso correspondiente.
- Servidor de autorización: Este servidor recibe las solicitudes de tokens de acceso del cliente y las emite una vez que el propietario del recurso se ha autenticado y ha dado su consentimiento. El servidor de autorización expone dos puntos de conexión: el punto de conexión de autorización, que maneja la autenticación interactiva y el consentimiento del usuario, y el punto de conexión de token, que está involucrado en una interacción de máquina a máquina.
- Servidor de recursos: Un servidor que protege los recursos del usuario y recibe las solicitudes de acceso del cliente. Acepta y valida un token de acceso del cliente y le devuelve los recursos adecuados.

¿Cómo funciona OAuth 2.0?

En el nivel más básico, antes de poder utilizar OAuth 2.0, el cliente debe adquirir sus propias credenciales, un id de cliente y un client secret, del servidor de autorización para identificarse y autenticarse al solicitar un token de acceso. Con OAuth 2.0, las solicitudes de acceso son iniciadas por el cliente, por ejemplo, una aplicación móvil, un sitio web, una aplicación de televisión inteligente, una aplicación de escritorio, etc. La solicitud, el intercambio y la respuesta de los tokens siguen el siguiente flujo general:

- El cliente solicita autorización (solicitud de autorización) al servidor de autorización, proporcionando el id y el client secret como identificación; también proporciona los ámbitos y un URI de extremo (URI de redireccionamiento) al que enviar el token de acceso o el código de autorización.
- El servidor de autorización autentica al cliente y verifica que los ámbitos solicitados están permitidos.
- El propietario del recurso interactúa con el servidor de autorización para conceder el acceso.
- El servidor de autorización redirige de vuelta al cliente con un código de autorización o un token de acceso, según el tipo de concesión, como se explicará en la siguiente sección. También puede devolverse un token de actualización.
- Con el token de acceso, el cliente solicita acceso al recurso desde el servidor de recursos.

Implementación de servicios en la nube (12-factor application)

La implementación de servicios en la nube siguiente el enfoque de 12-Factor, se basa en una serie de principio diseñados para crear aplicaciones modernas que sean escalables, mantenibles y portables. Estos principios son especialmente útiles para aplicaciones desplegadas en entornos de nube o contenedores, como los que se gestionan con Kubernetes o plataformas de nube como AWS, entre otras. La metodología "12-Factor" puede ser aplicada a aplicaciones escritas en cualquier lenguaje de programación, y cualquier combinación de 'backing services' (bases de datos, colas, memoria cache, etc).

1. Código base: Un código base sobre el que hacer el control de versiones y despliegues. Una aplicación de 12-factor se gestiona con un sistema de control de versiones como Git, etc.
2. Dependencias: declarar y aislar explícitamente las dependencias.
3. Configuración: guardar la configuración en el entorno. La configuración de una aplicación es todo lo que puede variar entre despliegues.
4. Backing services (Servicios de apoyo): cualquier recurso que la aplicación pueda consumir a través de la red como parte de su funcionamiento.
5. Construir, distribuir, ejecutar: separar completamente la etapa de construcción de la etapa de ejecución.

6. Procesos: ejecutar la aplicación como uno o más procesos sin estado en el entorno de ejecución.

7. Asignación de puertos: publicar servicios mediante asignación de puertos.

8. Concurrencia: escalar mediante el modelo de procesos. Los procesos de las aplicaciones 12-facor se inspiran en el modelo de procesos de unix para ejecutar demonios.

9. Desechabilidad: hacer el sistema más robusto intentado conseguir inicios rapidos y finalizaciones seguras.

10. Igualdad entre desarrollo y producción: mantener desarrollo, preproducción y producción tan parecidos como sea posible.

11. Historiales: tratar los historiales como una transmisión de eventos para permitir observar el comportamiento de la aplicación durante su ejecución.

12. Administración de procesos: ejecutar las tareas de gestión o administración como procesos que solo se ejecutan una vez.

Observaciones y Comentarios

La orquestación de servidores, Kubernetes, 12-Factor, Cloud Native, son enfoques o herramientas de desarrollo de software que han tenido un auge en estos tiempos ya que ayuda a la simplificación de tareas y a la automatización de las mismas, aunque siempre representan un desafío el poder aplicar estos enfoques, los beneficios que representan son suficientes para que las organizaciones utilicen estas nuevas tecnologías para el desarrollo del software.

Conclusiones

La orquestación de servidores ayuda a simplificar la administración, mejorar la eficiencia y reducir los errores humanos en la gestión de infraestructura apoyándose en herramientas como Kubernetes que es ampliamente utilizado en la industria para gestionar aplicaciones modernas basadas en microservicios y contenedores, proporcionando una solución robusta para los desafíos de escalabilidad y gestión en entornos de aplicaciones. La arquitectura de microservicios es especialmente útil para aplicaciones grandes y complejas que necesitan ser escalables, flexibles y mantenibles y el enfoque de 12-Factor es fundamental para construir aplicaciones modernas en la nube que sean robustas, fáciles de mantener y capaces de aprovechar al máximo la infraestructura de nube

Referencias

[1] *Orquestación en Servidores* [En linea]. Disponible en: https://forum.huawei.com/enterprise/es/orquestaci%25C3%25B3n-en-servidores/thread/671615996298936320-667212884846981120

[2] *¿Qué es Kubernetes?* [En linea]. Disponible en:
https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

[3] *¿Qué es Kubernetes?* [En linea]. Disponible en:
https://cloud.google.com/learn/what-is-kubernetes?hl=es-419

[4] *¿Qué son los microservicios?* [En linea]. Disponible en:
https://aws.amazon.com/es/microservices/

[5] *¿Qué son y para qué sirven los microservicios?* [En linea]. Disponible en:
https://www.redhat.com/es/topics/microservices

[6] *¿Qué es OAuth 2.0?* [En linea]. Disponible en:
https://auth0.com/es/intro-to-iam/what-is-oauth-2

[7] *The Twelve-Factor App* [En linea]. Disponible en:
https://12factor.net/es/

Repositorio git disponible en:
https://github.com/JaraizL/SEMINARIO-DE-TECNOLOG-AS.git