1. Demystifying Serverless Applications-ES

¿Qué es serverless?

#aprendizaje

Cuando alguien te pide desarrollar una solución, lo último que suele importarle es cómo funcionará la infraestructura. La verdad es que, incluso para los desarrolladores, lo más importante de la infraestructura es que simplemente funcione bien.

Considerando esta realidad, la posibilidad de contar con un proveedor cloud que gestione dinámicamente la asignación y el aprovisionamiento de servidores, delegando la infraestructura subyacente al proveedor, puede ser el mejor escenario posible.

Eso es lo que promete la arquitectura serverless: un modelo que podemos usar para construir y ejecutar aplicaciones y servicios sin tener que gestionar la infraestructura subyacente nosotros mismos. Este enfoque abstrae completamente la gestión del servidor, permitiendo que los desarrolladores se concentren en su código.

Hay muchas ventajas que podemos considerar al usar la computación serverless. El hecho de no tener que preocuparse por el escalado puede considerarse la principal. Además, el proveedor de soluciones cloud mantiene la confiabilidad y la seguridad del entorno. Sumado a eso, con este enfoque tienes la opción de pagar por uso, pagando únicamente por lo que consumes, lo que habilita un modelo de crecimiento sostenible.

Serverless también puede considerarse un buen enfoque para acelerar el desarrollo de software, ya que solo te enfocas en el código necesario para entregar el programa. Por otro lado, puede resultar difícil supervisar una cantidad considerable de funciones, por lo que esta organización debe manejarse bien para no generar problemas al crear una solución con muchas funciones.

Desde la introducción de serverless, se han creado distintos tipos de funciones. Estas funciones actúan como triggers que se usan para iniciar el procesamiento. En cuanto se activa la función, la ejecución puede realizarse en diferentes lenguajes de programación.

Veamos ahora si las funciones pueden considerarse microservices o no.

¿Es serverless una forma de entregar microservices?

Si buscas la definición de microservices, encontrarás el concepto de entregar una aplicación como componentes débilmente acoplados que representan la implementación de una capacidad de negocio. Puedes construir algo así con un par de funciones, así que sí: serverless es una forma de entregar microservices.

Algunos especialistas incluso consideran la arquitectura serverless una evolución de los microservices, ya que su foco es ofrecer escalabilidad en un entorno seguro, habilitando la posibilidad de que un conjunto de funciones sea desarrollado, probado y desplegado de forma independiente, lo que aporta gran flexibilidad a la arquitectura de software. Esa es exactamente la filosofía principal de los microservices.

Imaginemos, por ejemplo, un microservice responsable de autenticar usuarios. Puedes crear funciones específicas para el registro, el inicio de sesión y el restablecimiento de contraseñas. Considerando que este conjunto de funciones puede crearse en un único proyecto serverless, tienes tanto la flexibilidad de crear funciones separadas como la posibilidad de definir el propósito del microservice.

El proyecto serverless soportará de forma natural la integración con bases de datos, colas de mensajería, especificaciones OpenAPI y otras APIs, habilitando los patrones de diseño típicamente necesarios para una arquitectura de microservices robusta. También es importante mencionar que mantener los microservices aislados, pequeños y preferiblemente reutilizables es una buena práctica que vale la pena seguir.

Cómo presenta Microsoft Azure el serverless

Azure Functions nos da la oportunidad de potenciar aplicaciones usando múltiples lenguajes de programación, incluyendo C#, JavaScript, F#, Java y Python.

Una de las características más destacadas de Azure Functions es su integración fluida con otros servicios de Azure y APIs de terceros. Por ejemplo, puede conectarse fácilmente con distintas bases de datos de Azure (desde Azure SQL Server hasta Azure Cosmos DB), Azure Event Grid para arquitecturas basadas en eventos, y Azure Logic Apps para la automatización de flujos de trabajo. Esta conectividad simplifica el proceso de construir aplicaciones complejas de nivel empresarial que aprovechan múltiples servicios.

Con los años, las posibilidades de Azure Functions han evolucionado. Hoy podemos incluso gestionar flujos de trabajo con estado y operaciones de larga duración usando Azure Durable Functions. Con esto, puedes orquestar procesos complejos que pueden ejecutarse en múltiples ejecuciones de funciones.

Pero Microsoft no solo creó un entorno para codificar funciones. También creó un pipeline completo para desarrolladores, siguiendo el proceso DevSecOps que hoy se discute y utiliza ampliamente en soluciones empresariales. Los desarrolladores pueden usar herramientas como Azure Pipelines, GitHub Actions y otros servicios de CI/CD para automatizar el proceso de despliegue. También puedes monitorear y diagnosticar eventos en estas funciones usando Azure Monitor y Application Insights, que facilitan la resolución de problemas y la optimización.

La solución PaaS también habilita distintas configuraciones para ajustar la escalabilidad y la seguridad. Dependiendo del plan de hospedaje que elijas, tendrás diferentes oportunidades de escalado:

- **Consumption plan:** La opción más básica y económica para comenzar con Azure Functions. Ideal para cargas de trabajo orientadas a eventos con escalado automático.
- **Flex Consumption plan:** Ofrece escalado elástico y rápido, combinado con soporte para redes privadas (integración con VNet).
- **Dedicated plan (App Service plan):** Adecuado para funciones de larga duración y escenarios que requieren un rendimiento más predecible y asignación de recursos.
- **Azure Container Apps plan:** Una sólida opción para arquitecturas basadas en microservices que usan múltiples stacks tecnológicos o requieren mayor flexibilidad.
- **Premium plan:** Diseñado para escenarios de alto rendimiento con capacidad de escalado bajo demanda, con soporte para características avanzadas como VNet, tiempos de ejecución más largos e instancias pre-calentadas.

Estos planes varían según el comportamiento de escalado, el cold start, la posibilidad de usar una red virtual y, por supuesto, el pricing. El plan *Consumption* es exactamente la esencia de serverless: no tienes idea de dónde ni cómo se ejecuta tu código, y solo pagas por la ejecución del mismo. Por otro lado, al seleccionar los planes *App Service* o *Container Apps environment*, tendrás más control sobre el hardware y el consumo de recursos, lo que significa que obtienes la flexibilidad de Azure Functions en tu solución, junto con la gestión necesaria para aplicaciones de mayor escala.

> ⚠ **Warning**

Por razones de seguridad, no se recomienda exponer las funciones directamente al público. Puedes optar por entregarlas a través de un application gateway, como Azure Application Gateway, o usar Azure API Management como punto de entrada para las APIs que desarrolles con Azure Functions.

Los triggers disponibles en Azure Functions

La idea básica de Azure Functions es que cada función requiere un trigger para iniciar su ejecución. Una vez que el trigger se activa, la ejecución de tu código comenzará poco después. Sin embargo, el tiempo que tarda en iniciarse puede variar según el plan de hospedaje seleccionado. Por ejemplo, en el Consumption plan, las funciones pueden experimentar cold starts: un retraso que ocurre cuando la plataforma necesita inicializar los recursos. También es importante entender que la función puede activarse más de una vez al mismo tiempo, lo que permite la ejecución en paralelo.

Azure Functions ofrece una variedad de triggers que permiten a los desarrolladores ejecutar código en respuesta a distintos eventos. Estos son los más utilizados:

- **HTTP Trigger:** Permite que la función se ejecute mediante una solicitud HTTP. Es útil para crear APIs y webhooks, donde la función puede invocarse usando métodos HTTP estándar.
- **Timer Trigger:** Ejecuta la función según un horario basado en el modelo NCRONTAB. Es ideal para tareas que deben realizarse a intervalos regulares, como operaciones de limpieza, procesamiento de datos o envío de reportes periódicos. Es importante mencionar que la misma función con timer trigger no se ejecuta nuevamente hasta que finaliza su primera ejecución, lo que ayuda a prevenir ejecuciones superpuestas y posibles conflictos.
- **Blob Storage Trigger:** Ejecuta la función cuando se crea o actualiza un blob en un contenedor de Azure Blob Storage. Es útil para procesar o transformar archivos, como imágenes o logs, a medida que se cargan.
- **Queue Storage Trigger:** Ejecuta la función en respuesta a mensajes agregados a Azure Queue Storage. Es útil para construir sistemas de procesamiento en segundo plano escalables y confiables.
- **Event Grid Trigger:** Ejecuta la función en respuesta a eventos publicados en Azure Event Grid. Es útil para reaccionar ante eventos de distintos servicios de Azure, como la creación, modificación o eliminación de recursos.
- **Service Bus Trigger:** Ejecuta la función cuando se reciben mensajes en una cola o topic de Azure Service Bus. Es ideal para gestionar la mensajería entre aplicaciones y construir flujos de trabajo complejos.
- **Cosmos DB Trigger:** Ejecuta la función en respuesta a creaciones y actualizaciones en Azure Cosmos DB. Es útil para procesar cambios en los datos en tiempo real, como actualizar un índice de búsqueda o disparar procesamiento adicional.

Estos triggers ofrecen flexibilidad y escalabilidad, permitiendo a los desarrolladores construir aplicaciones orientadas a eventos que pueden responder a distintos tipos de eventos de forma fluida. Vale mencionar que existen otros triggers disponibles en Azure Functions, que analizaremos con más detalle en los próximos capítulos.

Relacionado

- 00-Practical serverless and microservices with csharp