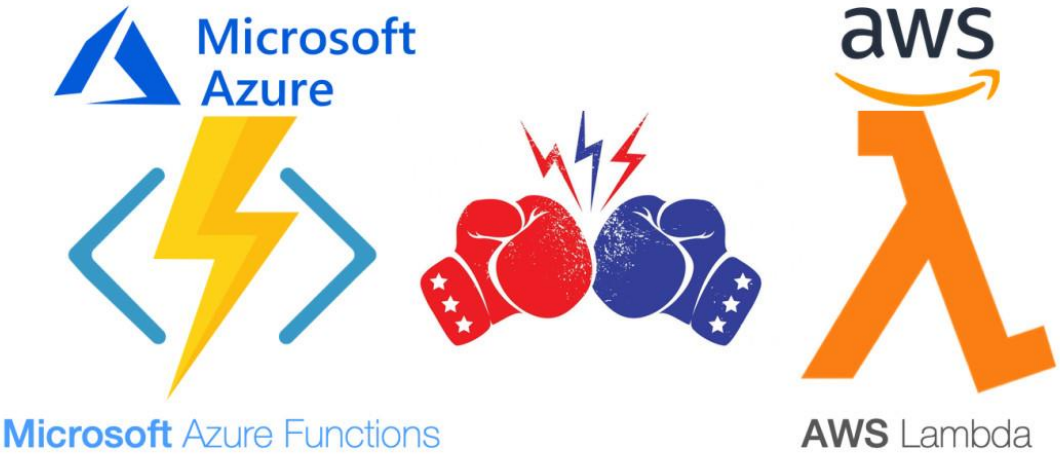
APLICACIONES DISTRIBUIDAS EN INTERNET

APLICACIONES SERVERLESS EN AWS O AZURE



Luis Díaz Madero
Benziane Mohammed Adel
Álvaro Lario Sánchez

Índice

Introducción

Las aplicaciones serverless son aquellas que se ejecutan sin tener que preocuparse por la gestión de servidores o infraestructura. En lugar de eso, los desarrolladores simplemente escriben su código y lo suben a una plataforma en la nube, que se encarga de ejecutar el código en un entorno aislado y escalar automáticamente según sea necesario.

Amazon Web Services (AWS) y Microsoft Azure son dos de las principales plataformas en la nube que ofrecen servicios de aplicaciones serverless. **AWS Lambda** es el servicio de aplicaciones serverless de Amazon, mientras que Azure Functions es la opción de Microsoft. Ambos servicios le permiten ejecutar código en respuesta a eventos y disparadores, lo que significa que solo cobran por el tiempo de ejecución real del código y no por el tiempo que los servidores están ociosos.

Ventajas de usar serverless

A continuación, se destacan algunos beneficios de usar este tipo de aplicaciones:

1. Escalabilidad: Estos servicios tienen la capacidad de adaptarse a los cambios según la demanda del cliente. Las arquitecturas serverless tienen la característica de ajustarse automáticamente y equilibrarse a las necesidades de los clientes sin necesidad de intervención manual. Esto permite una rápida adaptación.

2. Ahorro en costos: Solo se paga por lo que se usa. En este caso se paga por el tiempo que dura la ejecución en lugar de pagar por la instancia. Esto permite abaratar los costes si no se hace un gran uso.

3. Tolerancia a fallos y alta disponibilidad: Los servicios sin servidor ofrecen una alta disponibilidad y tolerancia a los fallos de forma predeterminada.

4. Administración de infraestructura: No es necesario administrar servidores y tampoco es necesario que los usuarios mantengan los sistemas operativos o la instalación y actualización de software. Estas tareas son responsabilidad del proveedor en la nube.

5. Eficiencia: La aplicación debe utilizar los recursos de manera eficiente, para minimizar los costos y maximizar el rendimiento.

6. Seguridad: la aplicación debe cumplir con todos los requisitos de seguridad y cumplir con los estándares de la industria.

7. Observabilidad: la aplicación debe tener métricas y herramientas de monitorización adecuadas para permitir una observación y depuración eficientes.

En general, este tipo de servicios serverless son ideales para aplicaciones de uso ocasional. Sin embargo, también hay consideraciones de diseño y arquitectura a tener en cuenta al usar estos servicios, como la gestión de dependencias y el manejo de errores y excepciones.

Servicios sin servidor central de AWS

Las aplicaciones sin servidor se suelen crear utilizando servicios completamente administrados como capas de informática. Algunos de estos servicios son AWS lambda, API Gateway, SQS, SNS, EventBridge. En la siguiente tabla se puede observar la categoría y el servicio de cada uno de ellos.

Categoría	Servicio
Informática	AWS Lambda
Proxy de la API	API Gateway
Mensajería e integración	SNS
	SQS
	EventBridge
Organización	Step Functions

AWS Lambda

AWS Lambda es un servicio de aplicaciones serverless de Amazon Web Services (AWS). Permite a los desarrolladores ejecutar código sin tener que preocuparse por la gestión de servidores o infraestructura. En su lugar, Permite a los desarrolladores subir su código a AWS Lambda, y este se encarga de ejecutar el código en un entorno aislado y escalar automáticamente según sea necesario. Este servicio está basado en eventos que le permite ejecutar código para prácticamente cualquier tipo de aplicación o servicio backend.

Los desarrolladores pueden elegir el lenguaje de programación que prefieran, ya que AWS Lambda es compatible con una amplia variedad de lenguajes, como Java, Python, C# y Node.js.

Amazon API Gateway

Este servicio hace más fácil a los desarrolladores la creación, publicaciones, mantenimiento y protección de las API. Con API Gateway es posible crear API WebSocket y API RESTful que

permiten aplicaciones de comunicación bidireccional en tiempo real. Este servicio permite caras de trabajo en contenedores, así como aplicaciones web.

Esta aplicación se para por las llamadas API que se reciben y por la cantidad de datos salientes transferidos.

La siguiente imagen sacada de la pagina de Amazon podemos ver el funcionamiento de este servicio:

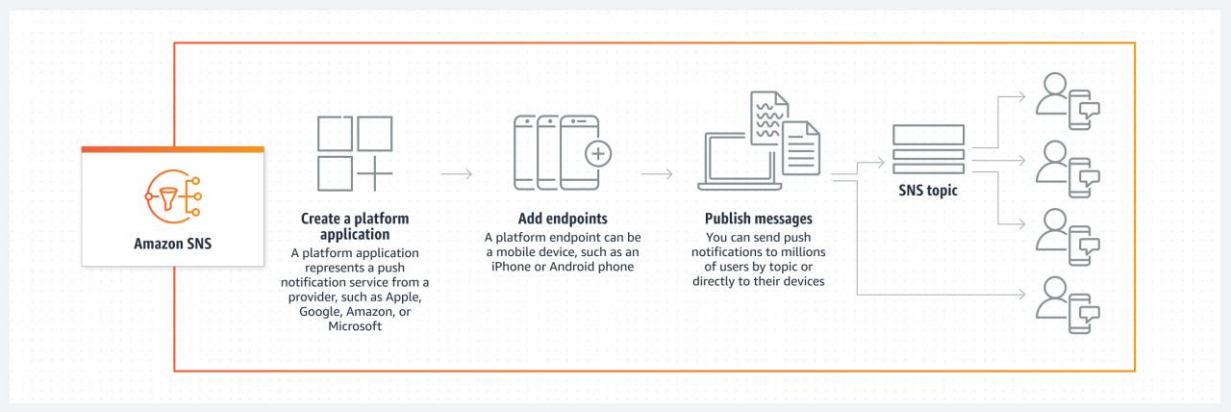
Amazon SNS

Amazon Simple Notification Service es un servicio de mensajería de publicación/suscripción que facilita el desacople y el ajuste de la escala de microservicios y los sistemas distribuidos.

Su funcionamiento es simple, puede mandar notificaciones de dos formas:

- A2A: Brinda mensajería de muchos a muchos y está basada en push entre sistemas distribuidos o aplicaciones controladas por eventos.
- A2P: Hace posible transferir mensajes a clientes con mensajes SMS y correos electrónicos.

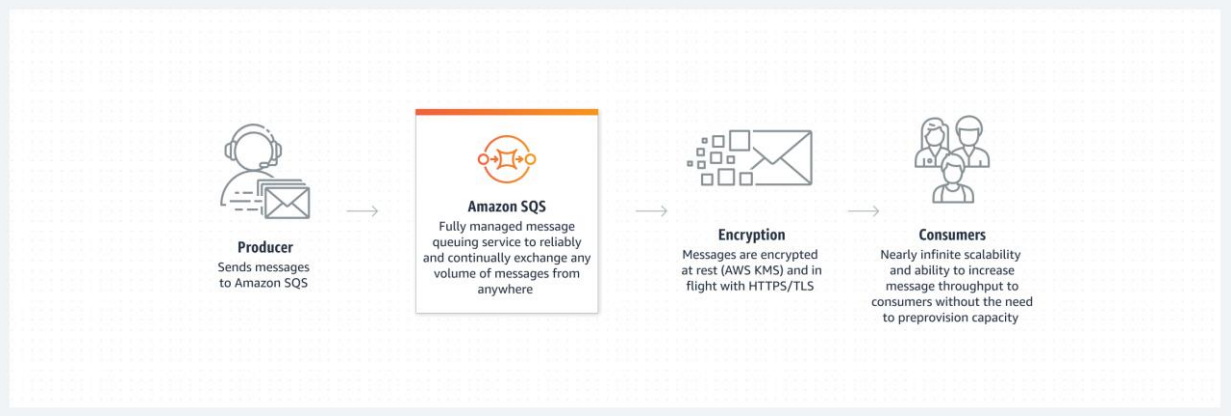
En la siguiente imagen de la página de Amazon se puede ver un esquema de los diferentes tipos de mensajes (publicación/suscripción, SMS, Notificaciones push móviles) En este caso voy a mostrar el esquema de notificaciones Push



Amazon SQS

Amazon Simple Queue Service se encarga de crear colas las cuales se pueden enviar, almacenar y recibir entre componentes de software, este servicio facilita el desacople y ajuste de microservicios, sistemas distribuidos o aplicaciones sin servidor.

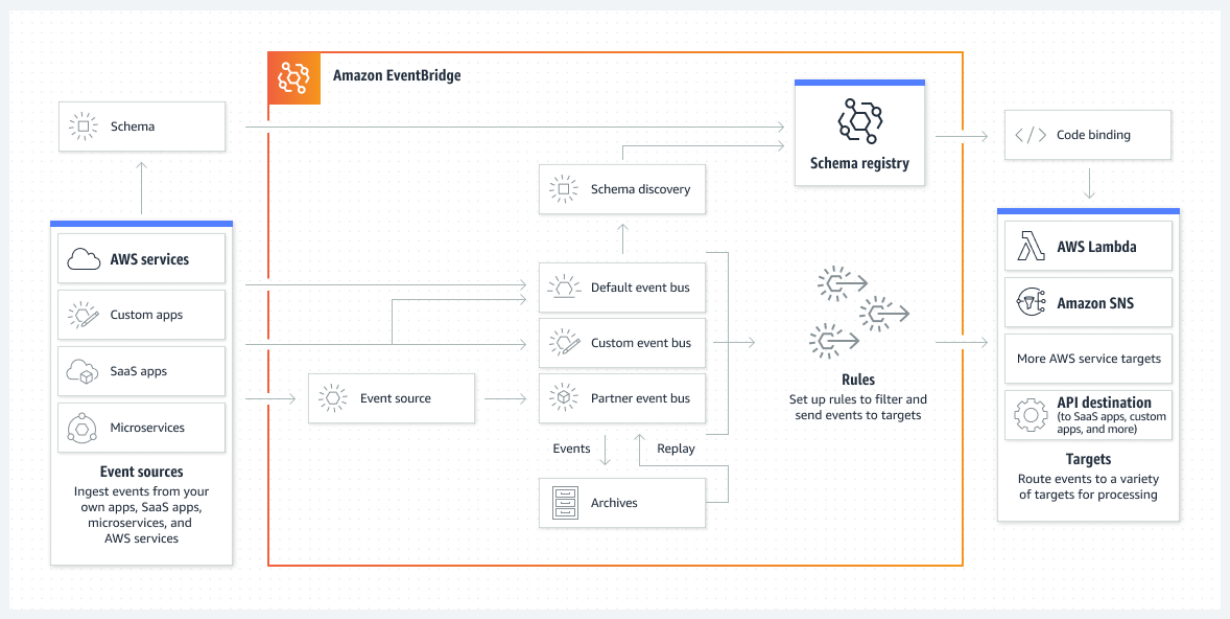
En la siguiente imagen, sacada de la página de amazon se muestra un esquema del funcionamiento de este servicio:



Amazon EventBridge

Este servicio permite que varios microservicios se registren en un bus de eventos sin servidor que facilita la conexión de aplicaciones entre si utilizando datos de sus propias aplicaciones, como (SaaS)

En la siguiente imagen de la página Amazon se puede ver un esquema del funcionamiento de este servicio.



AWS Step Functions

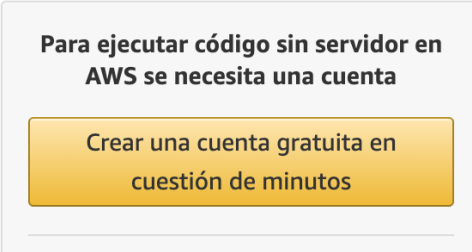
Este servicio permite a los usuarios diseñar y ejecutar flujos de trabajo de procesamiento de datos de manera sencilla y confiable.

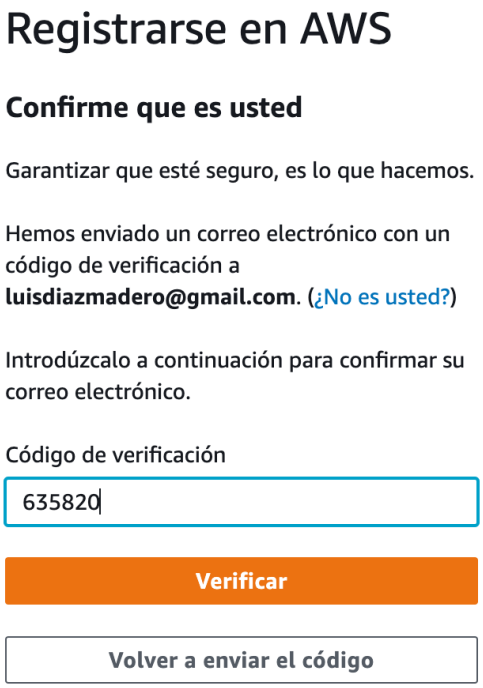
Una vez que se ha diseñado un flujo de trabajo es posible ejecutarlo en la nube mediante este servicio y gracias a esto es posible procesar grandes cantidades de datos de forma segura y escalable.

En la siguiente imagen de la página web de Amazon podemos observar un breve esquema del funcionamiento de este servicio.

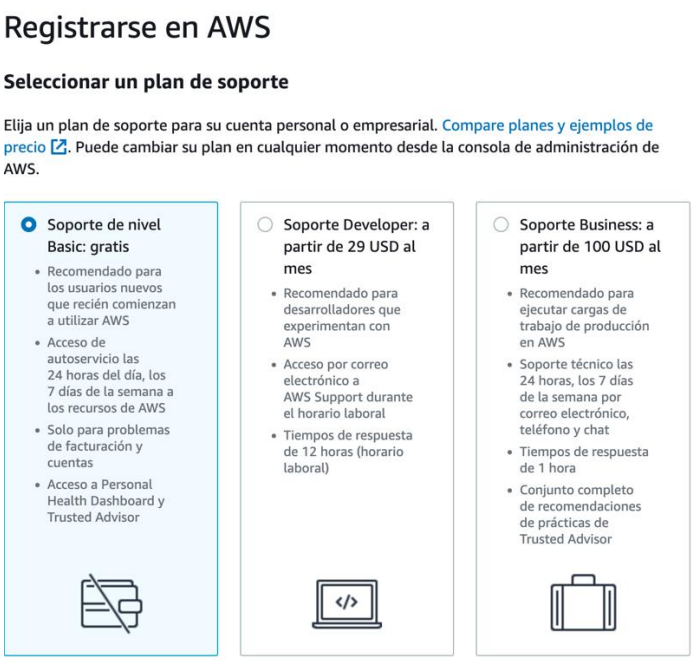
Ejemplo de Aplicación sin servidor AWS:

A continuación, se van a enseñar los conceptos básicos sobre la ejecución de código en AWS Lambda. Para empezar, es necesario crear una cuenta e introducir datos personales y bancarios (Este ejemplo es gratuito)

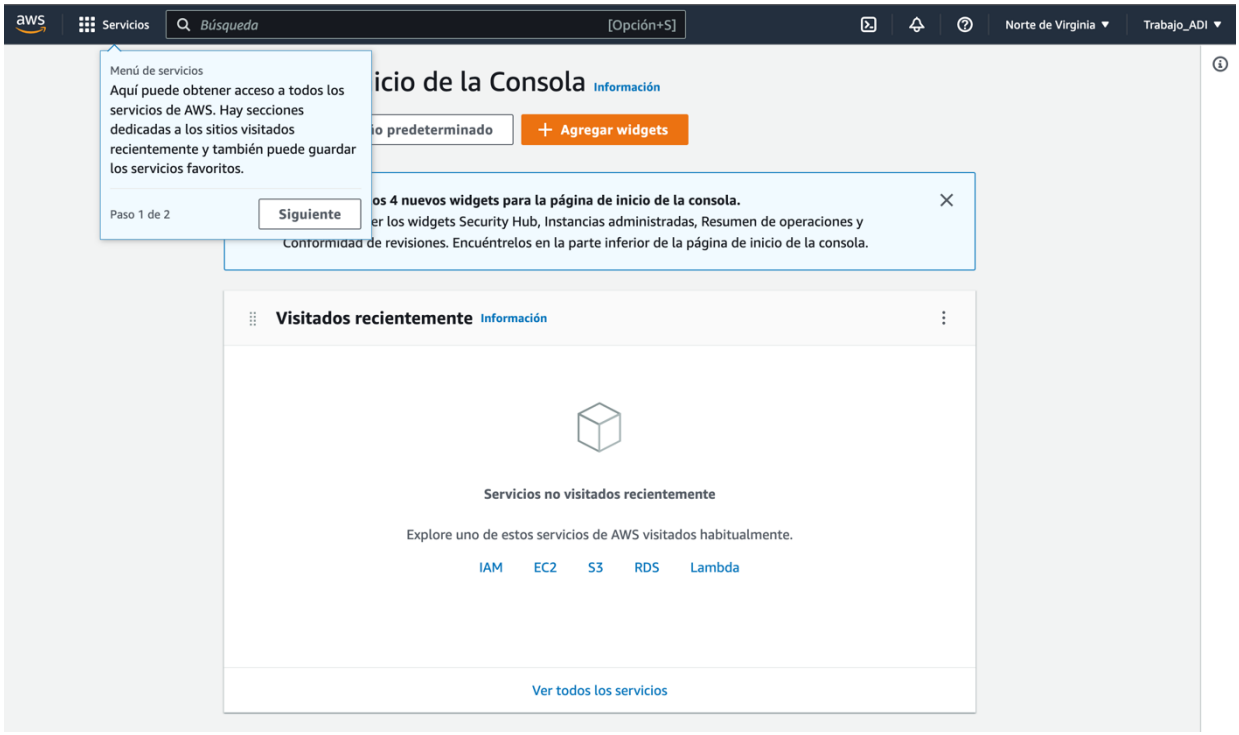




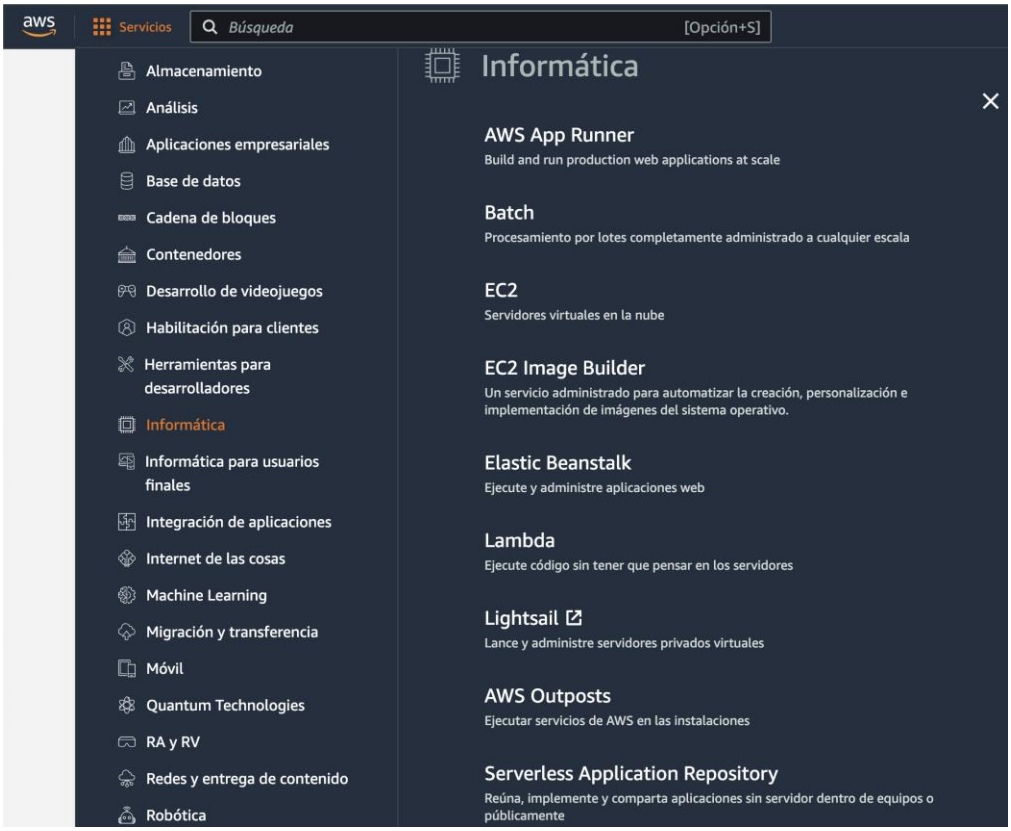
Una vez realizado el registro es necesario seleccionar un plan de soporte. En nuestro caso usaremos el soporte Basic que es gratuito y para ver el funcionamiento de este tipo de aplicaciones nos vale.



Después, accedemos a nuestra cuenta con el usuario raíz recién creada podremos observar la siguiente pantalla:

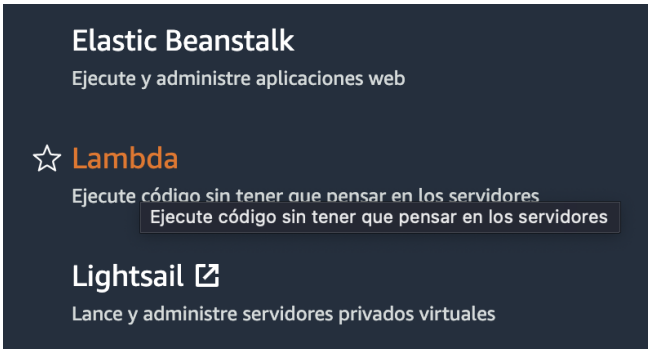


Esta es la página de inicio de la consola. A continuación, arriba a la derecha seleccionamos Servicios y se abrirá un menú desplegable donde deberemos acceder a Informática o Compute si está en inglés.



Seguidamente elegimos la opción Lambda como se puede ver en la siguiente imagen

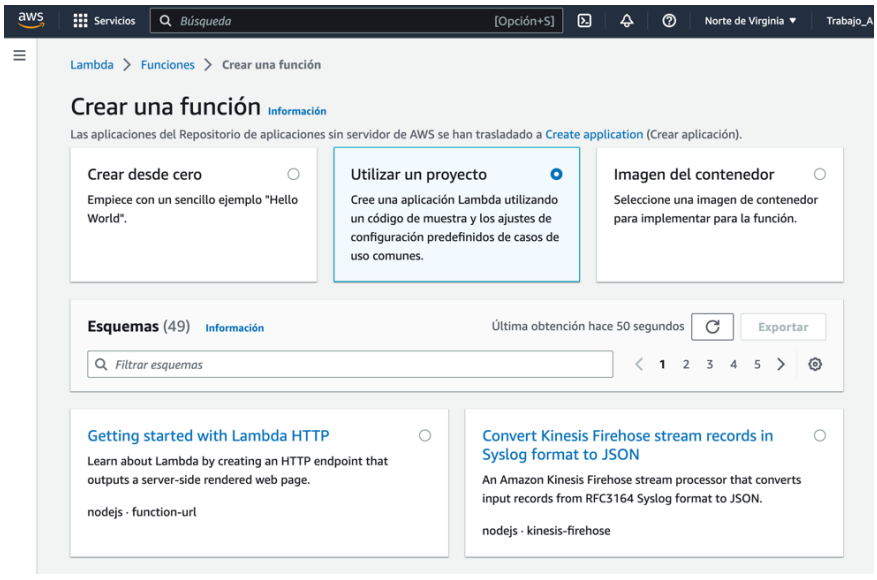
El siguiente paso es seleccionar un plano de lambda. Como se puede ver en la siguiente fotografía en la consola de AWS Lambda hay que seleccionar “Crear una función”

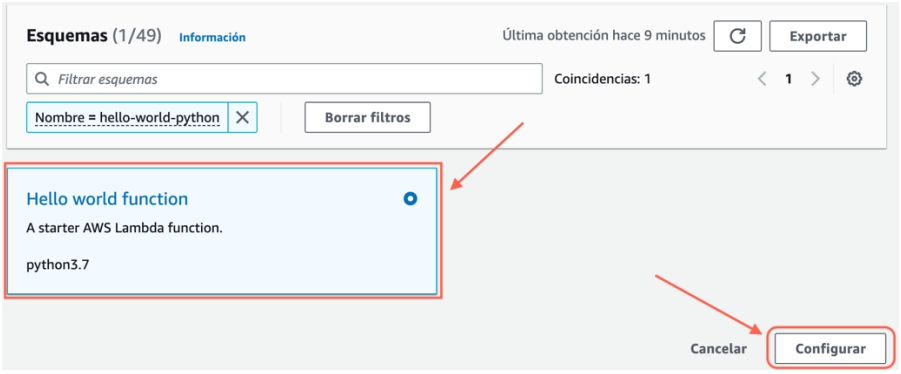


La consola solo mostrara esta página si no existe una función lambda creada. Si se han creado funciones se vera la página Lambda > Funciones.

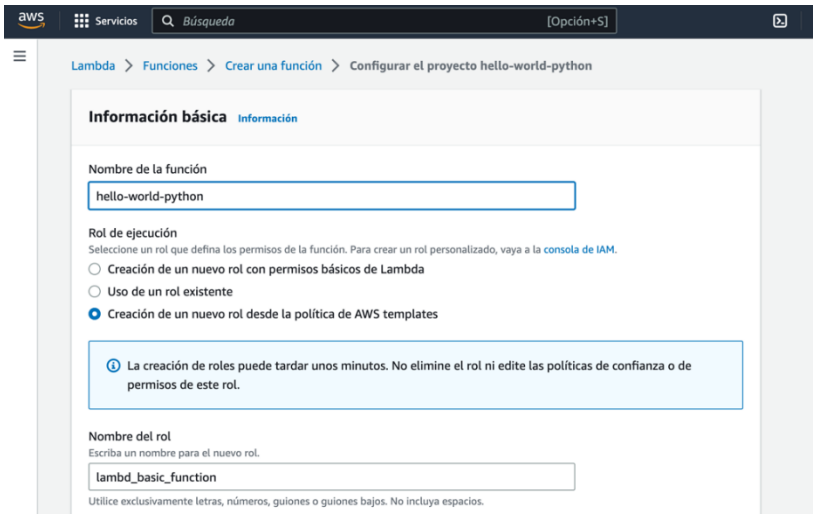
Una vez seleccionada la opción anterior aparecerá una página por pantalla donde deberemos seleccionar la opción “Utilizar un proyecto”

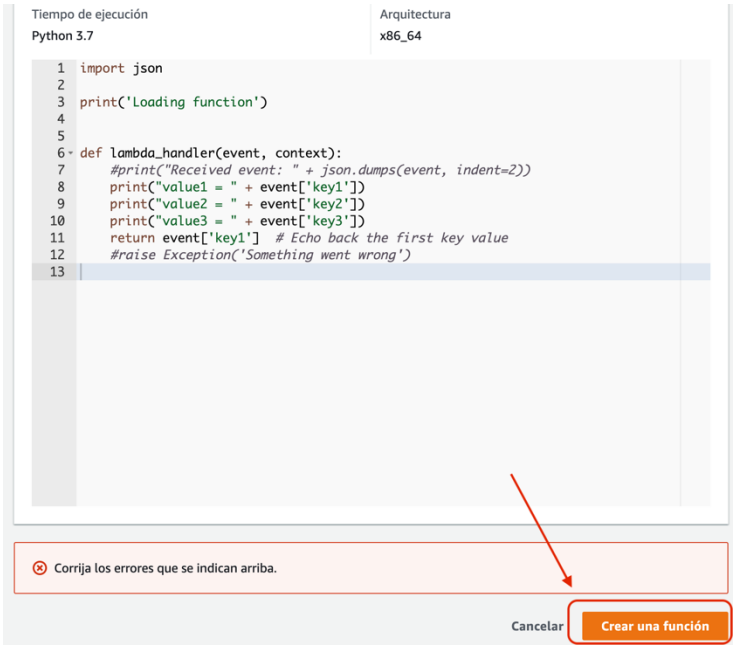
Abajo donde la lupa debemos buscar “Hello-World-python”





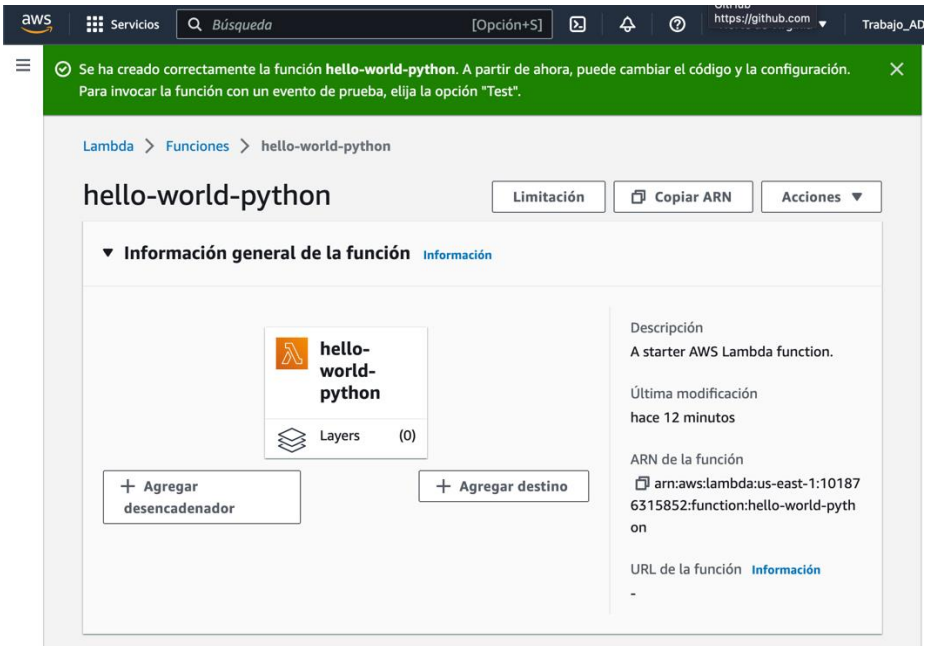
El siguiente paso es configurar la función lambda. En primer lugar, debemos poner un nombre a la función y asignar la casilla de “Creación de un nuevo rol desde la política de AWS templates”. En el nombre del rol escribiremos “lambda_basic_execution”

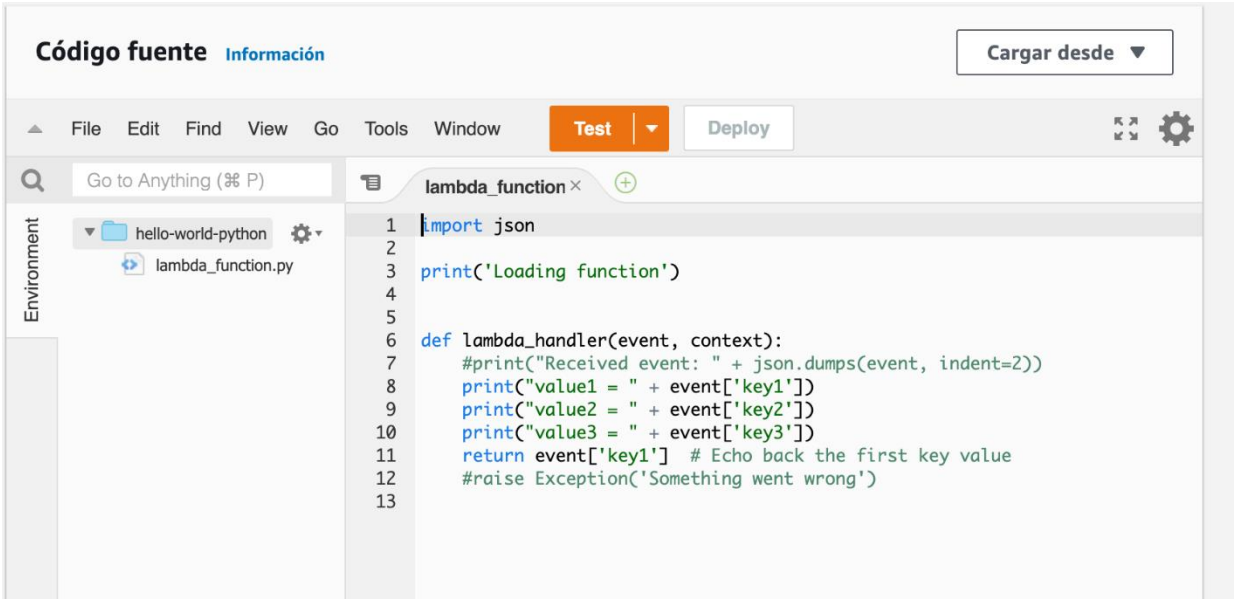




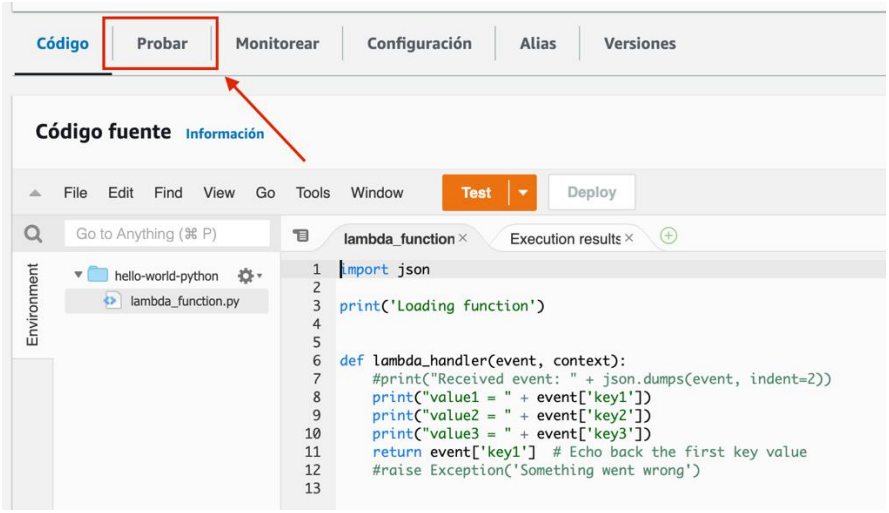
En la imagen de la izquierda podemos observar el código de nuestra aplicación, así como el lenguaje y la arquitectura, para continuar seleccionamos el botón “Crear una función”

En la siguiente fotografía se puede ver que efectivamente la aplicación se ha creado con éxito.

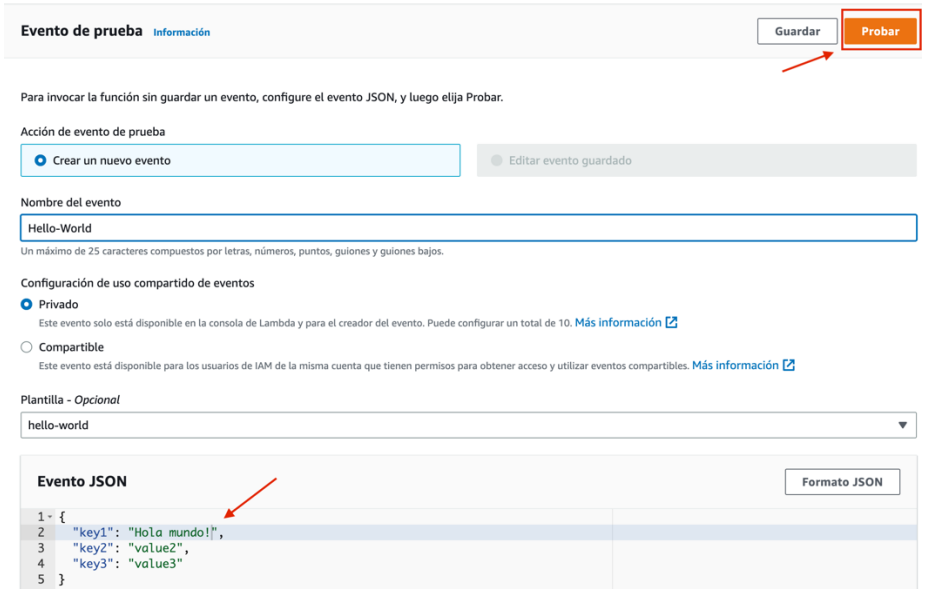


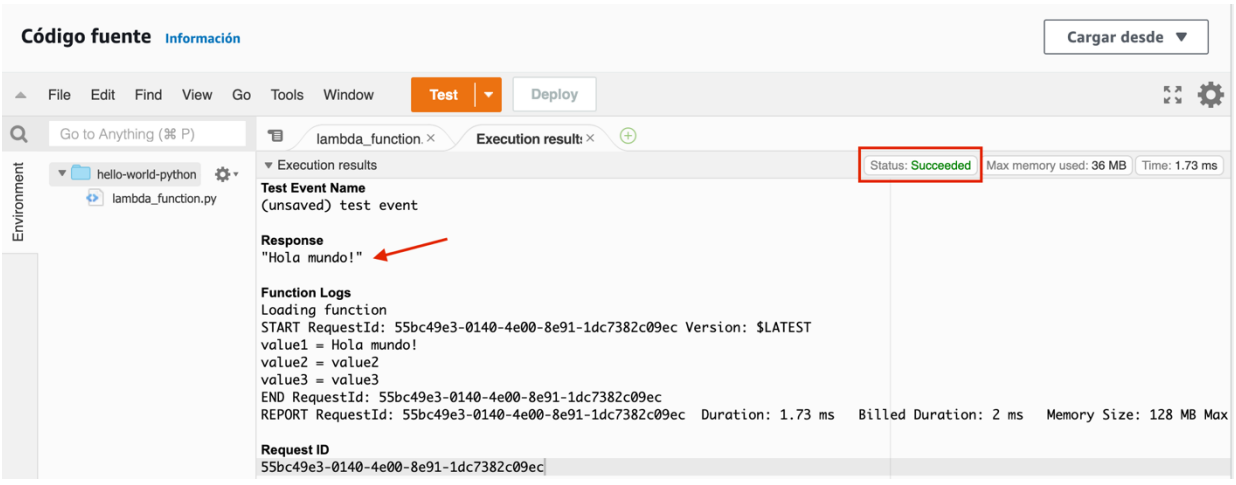


A continuación, vamos a invocar la función lambda y verificar los resultados, Vamos a la pestaña de probar.



En esta pestaña elegimos la opción de “Crear un nuevo evento” después el nombre del evento que deseemos ponerle, establecemos la configuración privada y luego en la pestaña “Evento JSON” cambiamos el valor de key1 por el mensaje “¡Hola mundo!”. Para terminar, apretaremos el botón “probar” como se puede ver en la siguiente imagen.





Finalmente, como se ve en la anterior fotografía ha sido un éxito y la respuesta es ¡Hola mundo!

Diseño impulsado por eventos

Como antes se ha explicado Amazon lambda es un servicio basado en eventos. Estos eventos se usan para activar y establecer comunicación entre servicios desacoplados.

En las arquitecturas basadas por eventos se destacan tres componentes clave:

- Procedimientos de eventos
- Enrutadores de eventos
- Consumidores de eventos

Disparadores y fuentes de eventos

Las funciones lambda también pueden generar sus propios eventos. Para esto existen múltiples opciones y de esta forma obtener gran flexibilidad para crear fuentes de eventos personalizados. Las principales fuentes de eventos pueden son:

- Depósitos de datos: Algunos servicios de bases de datos que nos proporciona Amazon, como puede ser:
    - Amazon S3: Amazon Simple Storage Service, permite almacenamiento de datos, seguridad y rendimiento.
    - Amazon DynamoDB: Servicio de base de datos NoSQL.
    - Amazon Kinesis: Facilita la recopilación, el procesamiento y análisis de los datos de streaming.
- Puntos de enlace emisores de eventos:
- Servicios de mensajería: Como los antes descritos, SQS o SNS.
- Acciones dentro de un repositorio: Por ejemplo, cuando se confirma un código a un repositorio AWS CodeCommit.

Implementación

Es recomendable asegurar que una modificación no interrumpa el contrato de servicio del cliente en una implementación en una arquitectura de microservicios. Si el dueño de la API realiza un cambio que rompe el contrato de servicio y el consumidor no está preparado para ello, se produce un fallo.

Para comprender el impacto de la implementación de cambios, es necesario conocer qué consumidores están utilizando la API. Puedes recopilar metadatos sobre el uso mediante claves de API, las cuales también pueden actuar como una forma de contrato en caso de realizar algún cambio en una API que cause una interrupción.

Para minimizar el impacto de los cambios de interrupción en una API, los clientes pueden clonar la API y dirigir a los clientes a subdominios diferentes (por ejemplo, v2.my-service.com) para garantizar que los consumidores existentes no se vean afectados. Este enfoque permite que los clientes implementen solo nuevos cambios en el nuevo contrato de servicio de la API, pero tiene compensaciones. Los clientes que sigan este enfoque deben mantener dos versiones de la API y enfrentar los costos generales del manejo y el aprovisionamiento de la infraestructura.

AWS Lambda frente a Azure Functions

Azure Functions es el equivalente AWS Lambda en la nube de Azure, podemos comparar los distintos servicios centrándonos en sus características y limitaciones. Una de sus principales diferencias se centra en los lenguajes de programación admitidos por cada servicio, en AWS Lambda podemos encontrar lenguajes comunes con Azure Functions como JavaScript, Java, Python, C#, PowerShell o F#. Este último carece de soporte para los lenguajes Go y Ruby.

Otra principal diferencia serían los modelos de programación, AWS Lambda tiene un modelo de programación sencillo. Una función recibe un objeto JSON como entrada y puede devolver otro JSON como salida. El tipo de evento define el esquema de esos objetos, que están documentados y definidos en SDK de lenguaje. Mientras que Azure Functions tiene un modelo más sofisticado basado en disparadores y enlaces. Un disparador es un evento que escucha la función. La función puede tener cualquier cantidad de enlaces de entrada y salida para extraer y/o enviar datos adicionales en el momento del procesamiento. Por ejemplo, una función desencadenada por HTTP también puede leer un documento de Azure Cosmos DB y enviar un mensaje de cola, todo ello de forma declarativa a través de la configuración de enlace.

Un punto importante sería la extensibilidad que facilita a los desarrolladores la adición de nuevas funciones y capacidades a las plataformas de software preexistentes. En el entorno de AWS Lambda se definen mecanismos de distribución para bibliotecas, tiempos de ejecución personalizados con el fin de admitir otros idiomas y otras dependencias. En cambio, Azure Functions habilita extensiones de enlaces abiertos para que la comunidad pueda crear nuevos tipos de enlaces y llevarlos a Function Apps.

El rendimiento es otro punto importante y en cual también existen diferencias. AWS Lambda siempre reserva una instancia separada para una única ejecución. Cada ejecución tiene su pool

exclusivo de memoria y ciclos de CPU. Por lo tanto, el rendimiento es totalmente predecible y estable. Azure Functions asigna varias ejecuciones simultáneas al mismo nodo virtual. Las ejecuciones que consumen muchos recursos pueden luchar por el conjunto de recursos compartidos, lo que perjudica el rendimiento general y el tiempo de procesamiento. Azure Functions ha mejorado significativamente en los últimos uno o dos años, pero Microsoft todavía se está poniendo al día.

Por último, hablaremos de la integración HTTP y la principal diferencia es que AWS Lambda Amazon introdujo la integración con Elastic Load Balancing (Alternativa a Amazon API Gateway) y el servicio cobra por horas de uso, en cambio Azure Functions viene con integración de punto de conexión HTTP lista para usar y no hay costo adicional para esta integración.

Como conclusión podemos decir que AWS Lambda y Azure Functions son servicios similares, pero el problema está en los detalles, y prácticamente todos los ángulos muestran algunas distinciones esenciales entre los dos. En definitiva, cada usuario se decantará por el servicio que más se adapte a su caso.

Bibliografía

OpenWebinars

AWS

Aws lambda

API Gateway

SNS

SQS

EventBridge

Step Functions

Primera aplicacion serverless en AWS

Introduction to Azure Functions