CF-BBA-03

Cloud Functions

David Victoria
(he/him)
Cloud Architect
Caylent

El espectro Serverless

Los microservicios le permiten descomponer para agilizar

Los microservicios eliminan la necesidad de concentrarse en la infraestructura como su diferenciador fundamental

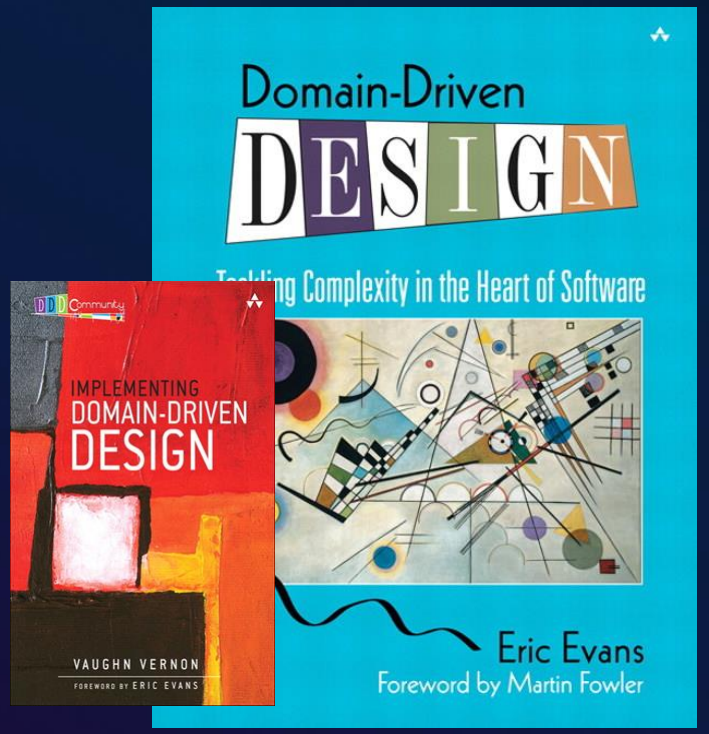
ENFOQUE EN INFRAESTRUCTURA

Instalación, configuración y administración
mi infraestructura informática es crítica
para lograr mis metas

ENFOQUE EN LAS APLICACIONES

Contar con un sistema estandarizado, flexible y entorno informático bajo demanda
es fundamental para lograr mis objetivos

Diseño basado en dominio (Domain Driven Design)



"Ubiquitous language" –Modelling the language of the business

Proporciona orientación sobre modelar dominios con entidades, objetos de valor, repositorios y servicios.

Top Tip!

Inicia leyendo desde el capítulo 11: Strategic Design vs Tactical Design

Los CSP ven la tecnología **serverless** como la arquitectura nativa de la nube

Sin aprovisionamiento sin gestión

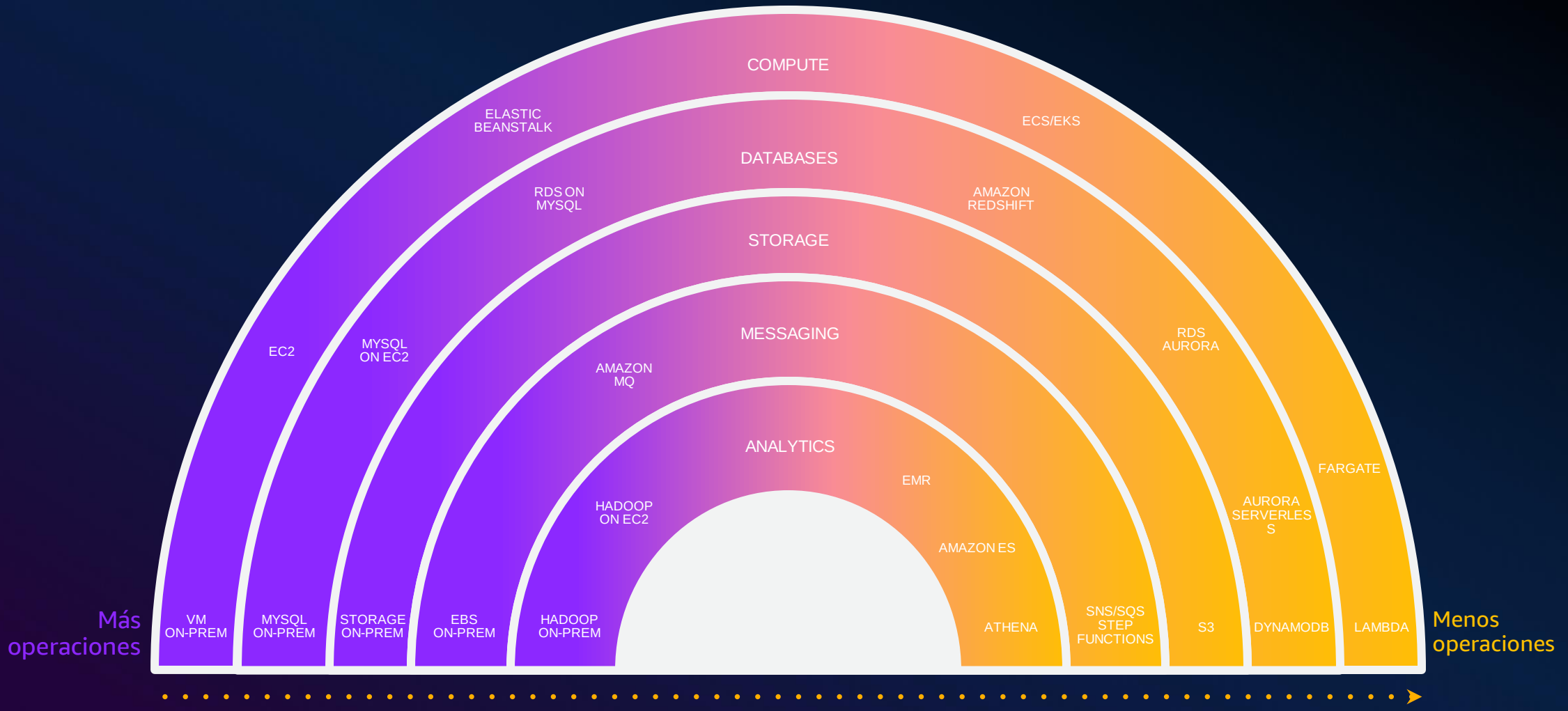
Escalado Automático

Pagar por uso

Altamente disponible y seguro

Source: Digital Rewrites The Rules Of Business, Forrester, February 2018

Serverless es una construcción operacional



Source: Digital Rewrites The Rules Of Business, Forrester, February 2018

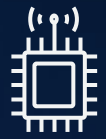
Ventajas de serverless

AGILITY

“Lo que nos llevó solo unos días construir con una solución sin servidor basada en AWS Lambda, nos hubiera llevado seis meses construirlo desde cero”.

—

Edmunds



ELASTICITY

“[Hemos] experimentado un tiempo de inactividad casi nulo y una degradación del rendimiento casi nula al atender entre 200 y 300 millones de solicitudes de OPI por mes”

—

Financial Engines

TOTAL COST EFFICIENCY

“El tamaño de nuestro equipo es la mitad de lo que normalmente se necesita para construir y operar un sitio de esta escala”.

—

Bustle

“Al usar AWS Lambda, hemos aumentado la rentabilidad en un factor de dos”.

—

FINRA

Serverless Foundations

Espectro de ofertas de AWS

On Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2

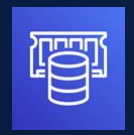
Microsoft SQL Server

Administrados

Amazon EMR

Amazon RDS

Amazon Elasticsearch



Amazon ElastiCache

Amazon Redshift

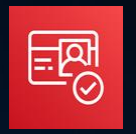
Amazon Neptune

Amazon SageMaker

Amazon MQ

Serverless

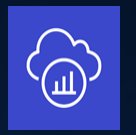
AWS Lambda



Amazon Cognito

Amazon Kinesis

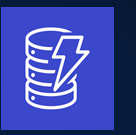
AWS StepFunctions



AWS X-Ray

Amazon Athena

Amazon S3



Amazon DynamoDB

Amazon SQS

Amazon API Gateway

Amazon CloudWatch

Amazon SNS

AWS Lambda: Serverless Computing

EVENT SOURCE

Cambios en el estado de los datos

Solicitudes a puntos finales

Cambios en el estado de los recursos

FUNCTION

Node.js
Python
Java
C#
Go
PowerShell
Ruby

SERVICES (ANYTHING)

Anatomía of a Lambda function

Handler() function

Función a ejecutar tras la invocación

Event object

Datos enviados durante la invocación de la función Lambda

Context object

Métodos disponibles para interactuar con la información de tiempo de ejecución (ID de solicitud, grupo de registros, etc.)

```
s3 = boto3.resource('s3')
app = App()

def lambda_handler(event, context):
        # do something
...
```

Precios detallados

Nivel gratuito (Free Tier)
1 millón de solicitudes y 400 000 GB-s de cómputo.
Cada mes, cada cliente.

- Compre tiempo de cómputo en incrementos de 1 ms
- Bajo cargo por solicitud
- Sin mínimos por hora, día o mes
- Sin cargos por dispositivo

Nunca pague por inactividad

Usa AWS Lambda

Trae tu propio código

- Node.js, Java, Python, C#, Go, Powershell, Ruby
- Traiga sus propias bibliotecas (incluso las nativas)

Modelo de recurso simple

- Seleccione la clasificación de potencia de 128 MB a 15 GB
- CPU y red asignadas proporcionalmente

Uso flexible

- Sincrónico o asincrónico
- Integrado con otros servicios de AWS

Autorización flexible

- Otorgue acceso seguro a recursos y VPC
- Control detallado para invocar sus funciones

Modelo de permisos de Lambda

Controles de seguridad detallados tanto para la ejecución como para la invocación:

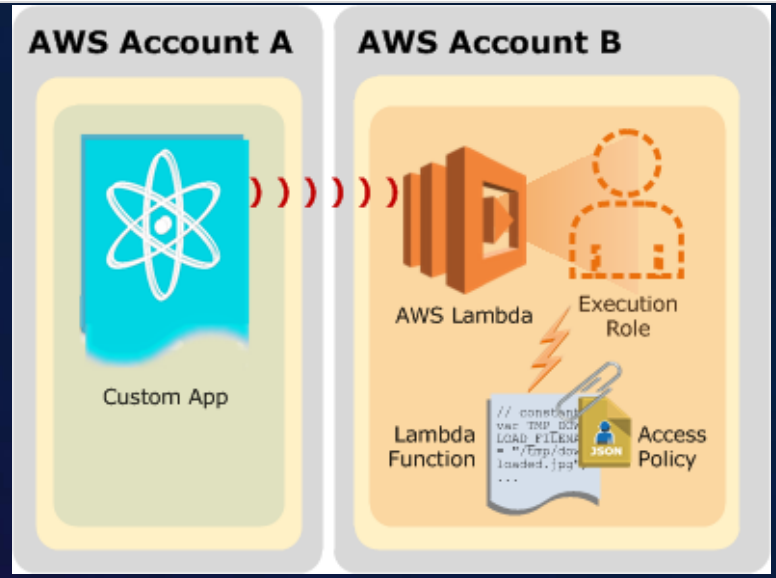
Políticas de ejecución:

- Defina a qué recursos de AWS/llamadas API puede acceder esta función a través de IAM
- Se utiliza en invocaciones de transmisión.
- P.ej. "La función Lambda A puede leer de los usuarios de tablas de DynamoDB"

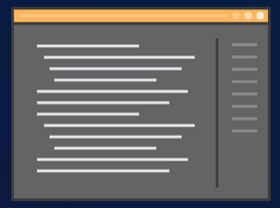
Políticas de funciones:

- Se utiliza para invocaciones sincronizadas y asincrónicas.
- P.ej. "Las acciones en el depósito X pueden invocar la función Lambda Z"
- Las políticas de recursos permiten el acceso entre cuentas

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```



Usando AWS Lambda



Creación de funciones

- Cloud9
- Editor WYSIWYG o cargar .zip empaquetado
- Complementos de terceros (Eclipse, Visual Studio)

Monitoreo y registro

- Métricas para solicitudes, errores y limitaciones
- Registros integrados en Amazon CloudWatch Logs
- Integración de x-ray

Modelo de programación

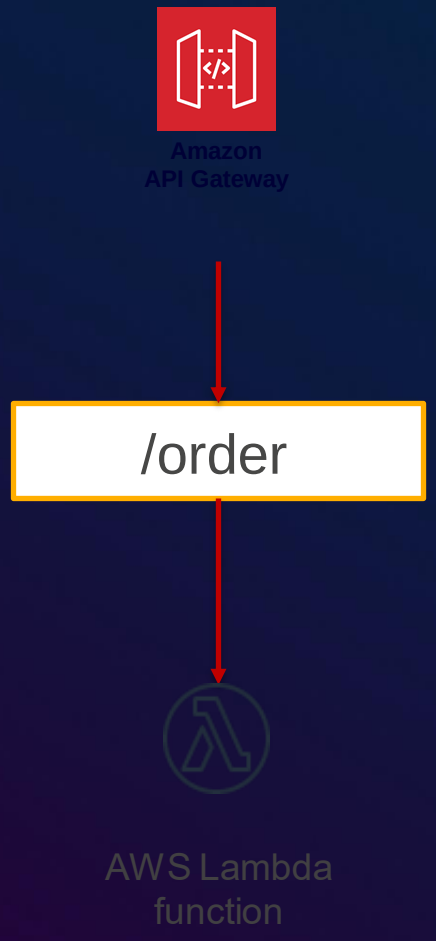
- Use procesos, subprocesos, /tmp, sockets normalmente
- AWS SDK integrado (Python y Node.js)

Stateless

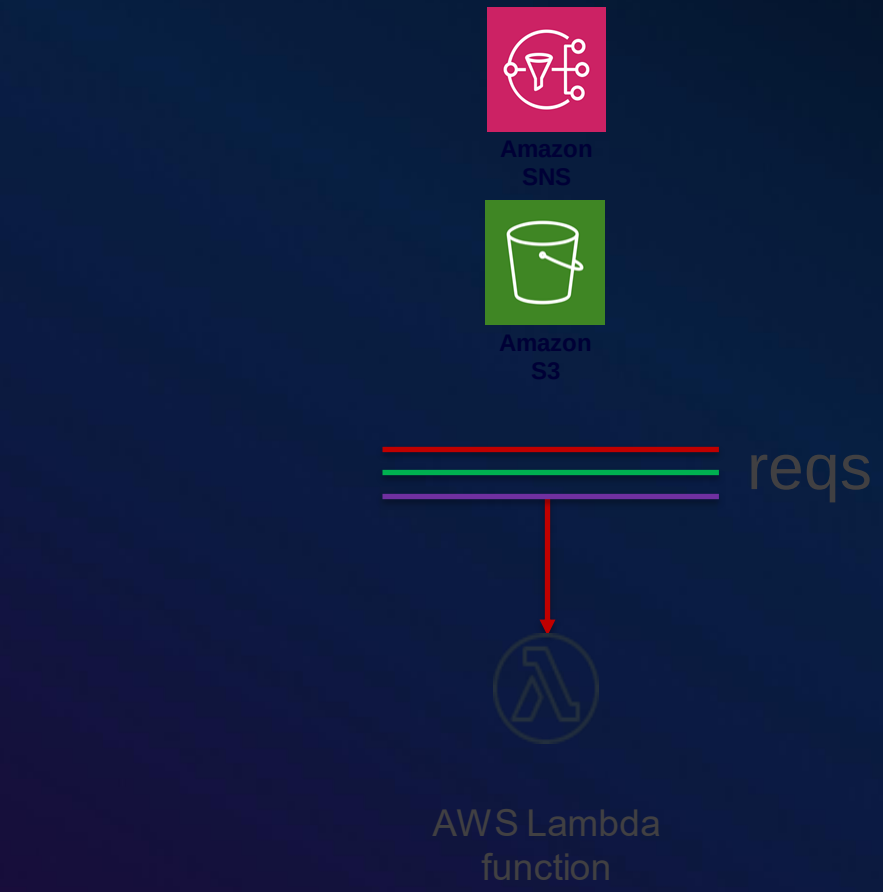
- Conservar datos usando almacenamiento externo
- Sin afinidad o acceso a la infraestructura subyacente

Modelos de ejecución de Lambda

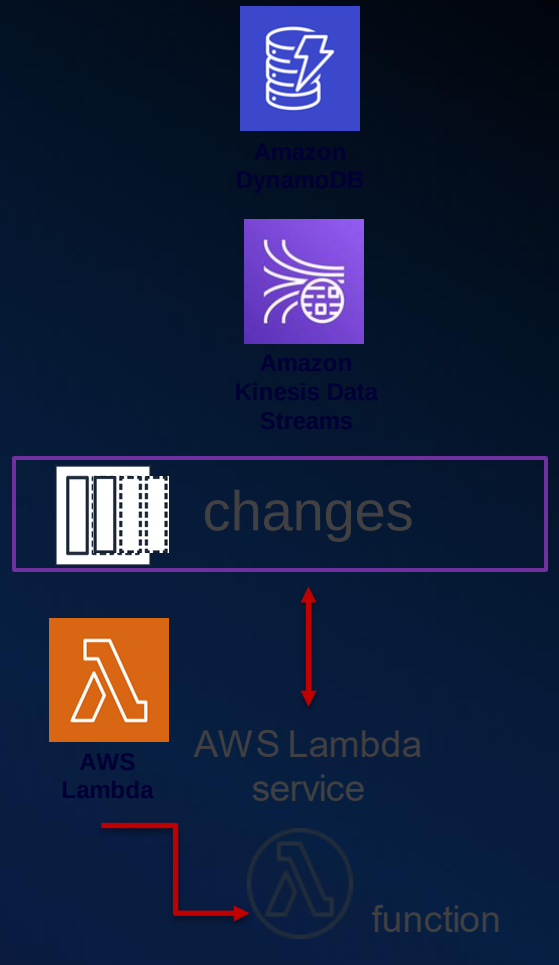
Synchronous (push)



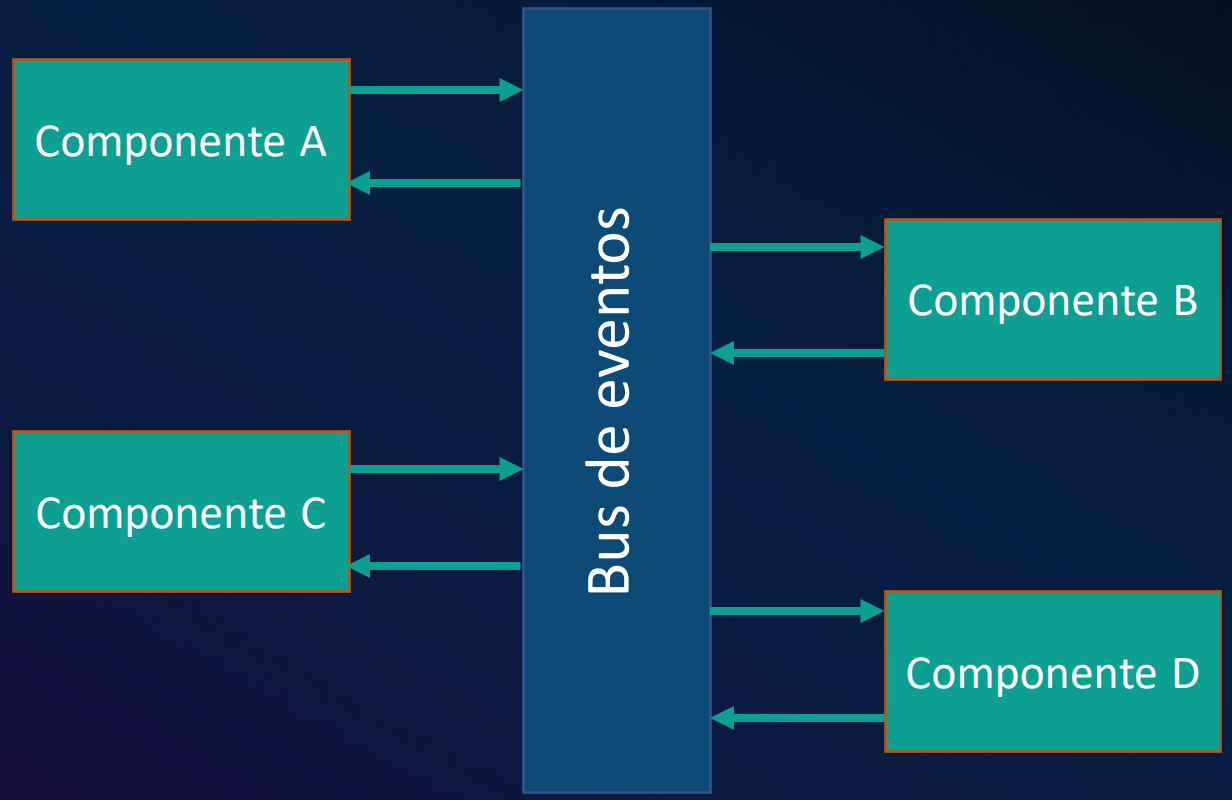
Asynchronous (event)



Poll-based



Arquitecturas event-drive



Arquitecturas event-drive

Una arquitectura impulsada por eventos utiliza eventos para desencadenar y comunicarse entre servicios desacoplados y es común en aplicaciones modernas creadas con microservicios. Un evento es un cambio de estado o una actualización, como un item que se coloca en un carrito de compras en un sitio web de comercio electrónico. Los eventos pueden incluir el estado (el artículo comprado, su precio y una dirección de entrega) o los eventos pueden ser identificadores (una notificación de que se envió un pedido).

Las arquitecturas impulsadas por eventos tienen tres componentes clave: ***productores de eventos***, ***enrutadores de eventos*** y ***consumidores de eventos***. Un productor publica un evento en el enrutador, que filtra y envía los eventos a los consumidores. Los servicios al productor y los servicios al consumidor están desacoplados, lo que permite escalarlos, actualizarlos y desplegarlos de forma independiente.

Beneficios de arquitecturas event-drive

Escalabilidad y fallos de forma independiente

Al desacoplar los servicios, solo conocen el enrutador de eventos, no entre ellos. Esto significa que los servicios son interoperables, pero si un servicio falla, el resto seguirá funcionando. El enrutador de eventos actúa como un búfer elástico que se adapta a los aumentos repentinos de las cargas de trabajo.

Audite con facilidad

Un enrutador de eventos actúa como una ubicación centralizada para auditar su aplicación y definir políticas. Estas políticas pueden restringir quién puede publicar y suscribirse a un enrutador y controlar qué usuarios y recursos tienen permiso para acceder a sus datos. También puede cifrar sus eventos tanto en tránsito como en reposo.

Beneficios de arquitecturas event-drive

Desarrollar con agilidad

Ya no es necesario escribir código personalizado para sondear, filtrar y enrutar eventos; el enrutador de eventos filtrará y enviará eventos automáticamente a los consumidores. El enrutador también elimina la necesidad de una gran coordinación entre los servicios del productor y el consumidor, lo que acelera su proceso de desarrollo.

Reducir costos

Las arquitecturas impulsadas por eventos están basadas en push, por lo que todo sucede bajo demanda a medida que el evento se presenta en el enrutador. De esta manera, no pagará por un sondeo continuo para verificar si hay un evento. Esto significa menos consumo de ancho de banda de red, menos utilización de CPU, menos capacidad de flota inactiva y menos apretones de manos SSL / TLS.

Cuando usar arquitecturas event-drive

Replicación de datos entre cuentas y regiones

Puede utilizar una arquitectura impulsada por eventos para coordinar los sistemas entre los equipos que operan y se implementan en diferentes regiones y cuentas. Al utilizar un enrutador de eventos para transferir datos entre sistemas, puede desarrollar, escalar e implementar servicios independientemente de otros equipos.

Procesamiento en paralelo y en abanico

Si tiene muchos sistemas que necesitan operar en respuesta a un evento, puede usar una arquitectura impulsada por eventos para distribuir el evento sin tener que escribir código personalizado para enviarlo a cada consumidor. El enrutador enviará el evento a los sistemas, cada uno de los cuales puede procesar el evento en paralelo con un propósito diferente.

Cuando usar arquitecturas event-drive

Supervisión y alerta del estado de los recursos

En lugar de verificar continuamente sus recursos, puede usar una arquitectura impulsada por eventos para monitorear y recibir alertas sobre cualquier anomalía, cambio y actualización. Estos recursos pueden incluir depósitos de almacenamiento, tablas de bases de datos, funciones sin servidor, nodos de cómputo y más.

Integración de sistemas heterogéneos

Si tiene sistemas que se ejecutan en diferentes pilas, puede utilizar una arquitectura dirigida por eventos para compartir información entre ellos sin acoplamiento. El enrutador de eventos establece indirección e interoperabilidad entre los sistemas, de modo que puedan intercambiar mensajes y datos sin dejar de ser agnósticos.

Demo

¡Gracias!

David Victoria

@vikomex

linkedin.com/in/vikomex

hi@davidvictoria.com