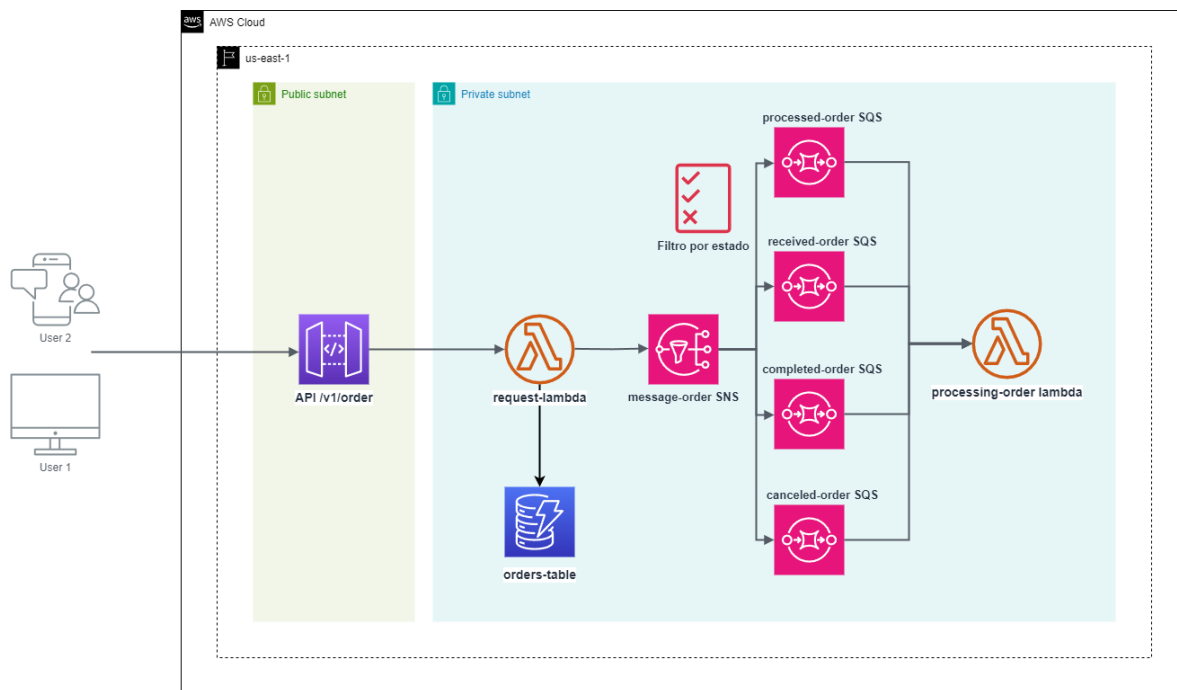


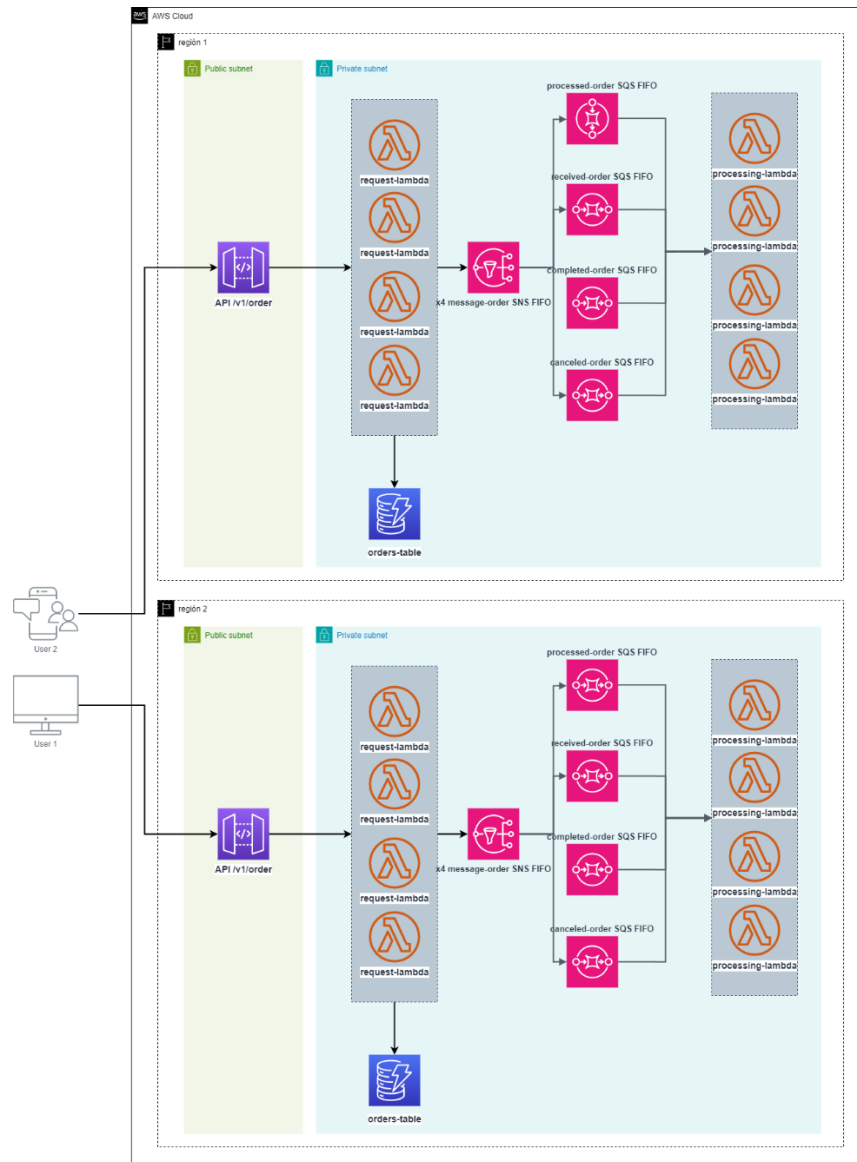
Arquitectura propuesta para la solución



- Se utilizó API Gateway para recibir las solicitudes del cliente.
- Se creó una lambda request para recibir las solicitudes del api, guardar en la base de datos y enviar un mensaje al sistema de colas, se usó lambda por su facilidad de instalación y altamente escalable para un proceso de bajo esfuerzo en procesamiento.
- Se utilizó base de datos dynamo para almacenar la información por su facilidad de instalación y altamente escalable, dado que no se tiene una fuerte relación entre las entidades de los datos es una buena opción.
- Para el sistema de colas, se implementó una integración entre sns para recibir los mensajes de las solicitudes y aplicar una política de filtros por atributos del mensaje y enviar una cola específica para cada estado de solicitud, generando desacople en desarrollo y generando escalabilidad de infraestructura en caso de requerirse más estados.
- Finalmente se creó la lambda processing para recibir los mensajes de cada cola y procesarlos.

Propuestas de escalamiento

Alternativa #1



En caso de que se incremente el volumen de las solicitudes que lleguen al límite de concurrencia de 1000 lambdas, se propone escalar a crear una lambda por cada estado y tener mayor capacidad de concurrencia. Si aún así se requiere más capacidad, se puede optar por el multi región y replicar la infraestructura en otra región cercana geográficamente y lograr una doble capacidad de la infraestructura inicial.

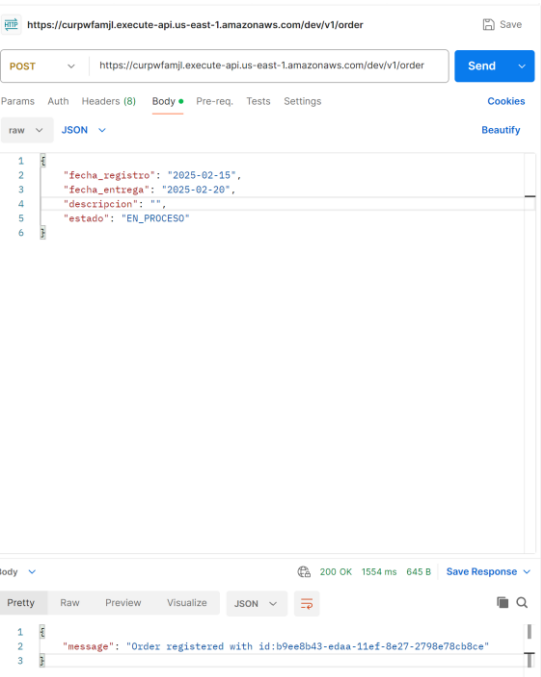
Para garantizar el orden de llegada de las solicitudes, se propone configurar los temas de sns y colas sqs de estándar a **FIFO**.

Alternativa #2

Como otra alternativa, se propone reemplazar las lambdas por kubernetes donde se puede alcanzar un límite hasta 50.000 pods aprovisionando la cpu y memoria necesaria para alcanzar la máxima transaccionalidad deseada.

Evidencias de funcionamiento

Respuesta exitosa del API



Ejecución de lambda

[tro](#)
[aws/lambda/work-order-backend-dev-request-lambda](#)
2025/02/18/[\$LATEST]538807d393084fd5a644cd3583b86a1

Eventos de registro

Puede utilizar la barra de filtros a continuación para buscar y hacer coincidir términos, frases o valores en sus eventos de registro.
 [Más información sobre los patrones de filtro](#)

Borrar

1m

30m

1h

12h

Personalizado

Zona horaria local

Mostrar

►

Marca temporal

►

Mensaje

No hay eventos antiguos en este momento. [Volver a intentar](#)

►	2025-02-17T22:44:58.137-05:00	INIT_START Runtime Version: python:3.10.v46 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:732ad977322a3d3b48316f227272d490c5a1f1898f80a758b6a1de53725c9e97d
►	2025-02-17T22:44:59.138-05:00	START RequestId: 35f97c96-bc6b-4b93-9c0d-b05648fc31d7 Version: \$LATEST
►	2025-02-17T22:44:59.139-05:00	{'resource': '/[proxy+]', 'path': '/v1/order', 'httpMethod': 'POST', 'headers': {'Accept': '**/*', 'Accept-Encoding': 'gzip, deflate, br', 'CloudFront-Forwarded-Proto': 'https'}}
►	2025-02-17T22:44:59.248-05:00	{'MessageId': '8675fa67-4f07-51ee-b80f-9a494b7c58ce', 'ResponseMetadata': {'RequestId': '6afcf7643-7609-5c12-a202-166d7272a532', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn'}}
►	2025-02-17T22:44:59.251-05:00	END RequestId: 35f97c96-bc6b-4b93-9c0d-b05648fc31d7
►	2025-02-17T22:44:59.251-05:00	REPORT RequestId: 35f97c96-bc6b-4b93-9c0d-b05648fc31d7 Duration: 112.43 ms Billed Duration: 113 ms Memory Size: 1024 MB Max Memory Used: 90 MB Init Duration: 999.76 ms

No hay eventos recientes en este momento. [Reintentar](#)

orders-table-dev

✔ Completado. Unidades de capacidad de lectura consumidas: 0.5

Llamado a lambda processing con evento sqs

[illegible]