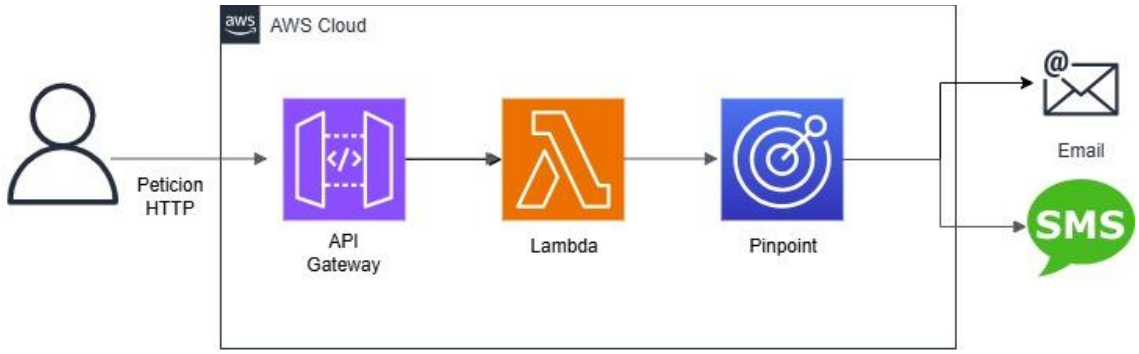
Sistema Serverless para Envío de Notificaciones SMS

El objetivo de esta implementación es proporcionar a la empresa o usuario una solución eficiente y escalable para enviar notificaciones SMS, por ejemplo, a sus clientes. Este sistema se basa en varios servicios de AWS que trabajan juntos de la siguiente manera

Utilizamos Amazon API Gateway para recibir solicitudes HTTP, las cuales desencadenan una función AWS Lambda. Esta función Lambda interactúa con Amazon Pinpoint para enviar mensajes SMS. Este enfoque permite una arquitectura serverless eficiente, donde API Gateway maneja las solicitudes entrantes, Lambda ejecuta la lógica de negocio y Pinpoint gestiona el envío de SMS, eliminando la necesidad de servidores dedicados y reduciendo costos operativos.

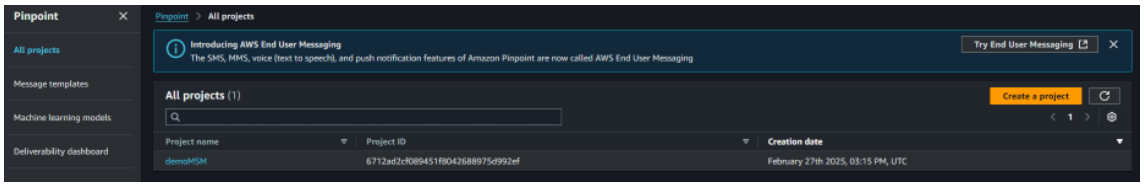
Arquitectura



Amazon Pinpoint

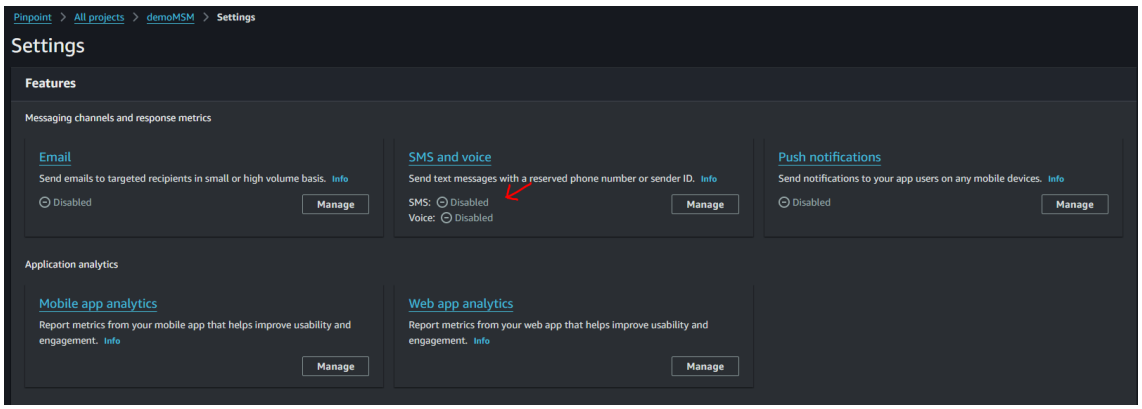
Amazon Pinpoint es un servicio de AWS que permite a las empresas enviar mensajes personalizados y segmentados a través de múltiples canales, como correo electrónico, SMS, notificaciones push y más. Es ideal para campañas de marketing, notificaciones de aplicaciones y comunicaciones con los clientes. Es importante tener en cuenta que esto es una demo y usando cuenta Sandbox de Amazon Pinpoint para SMS, solo puedes enviar mensajes a números de teléfono verificados. Además, tu límite de gasto mensual está fijado en $1 y puedes tener hasta 10 números de teléfono verificados por región de AWS. Para eliminar estas restricciones, puedes solicitar un aumento de cuota a través del soporte de AWS.

Creamos un nuevo proyecto:

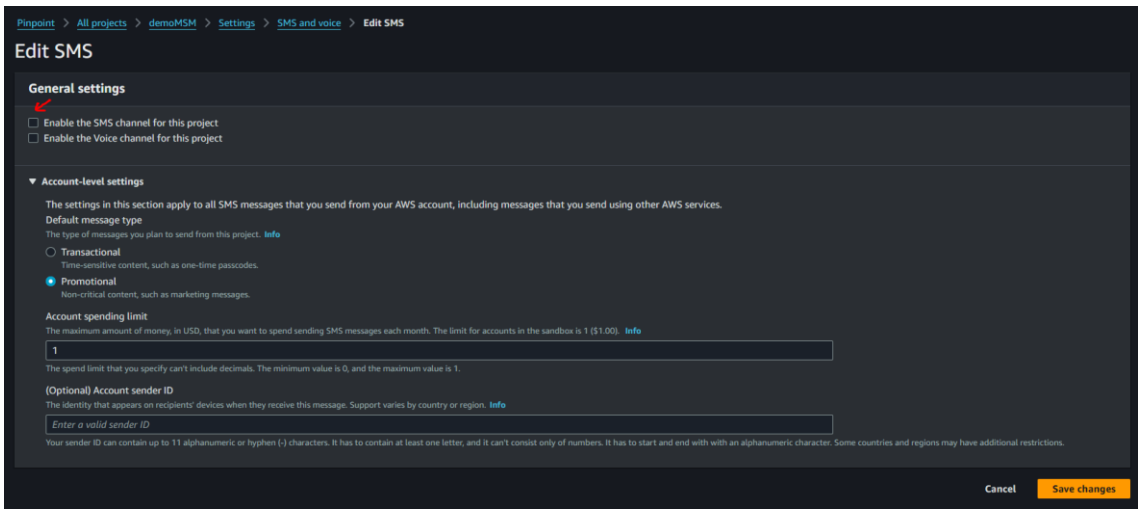


El “Project ID” es importante, puesto que lo necesitaremos para identificar nuestro servicio posteriormente en la lambda.

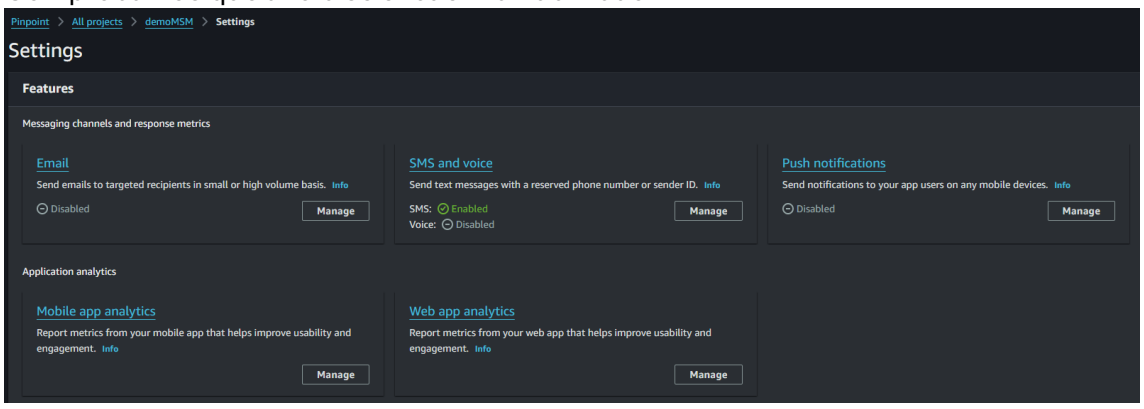
A la hora de configurar Amazon Pinpoint por defecto los SMS y demás servicios están deshabilitados.



Debemos habilitarlo para que funcione correctamente. Por tanto, en Settings/SMS > Edit activamos el servicio de SMS:



Comprobamos que ahora se encuentra habilitado:

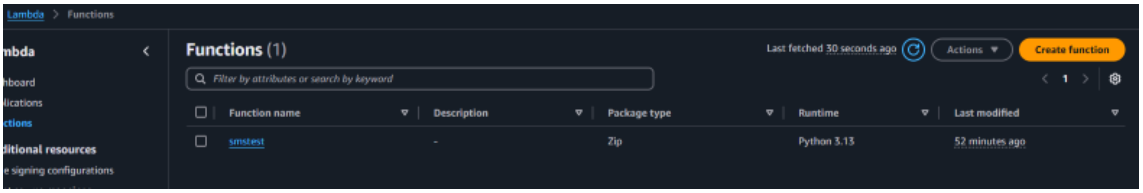


Lambda

Hemos optado por AWS Lambda para ejecutar scripts bajo demanda porque ofrece escalabilidad automática, costos reducidos al pagar solo por el tiempo de ejecución, integración sencilla con otros servicios de AWS y despliegue rápido. Esto elimina la

necesidad de gestionar servidores, permitiéndonos tener un sistema completamente serverless y evitar los costos y el mantenimiento de una instancia dedicada 24/7.

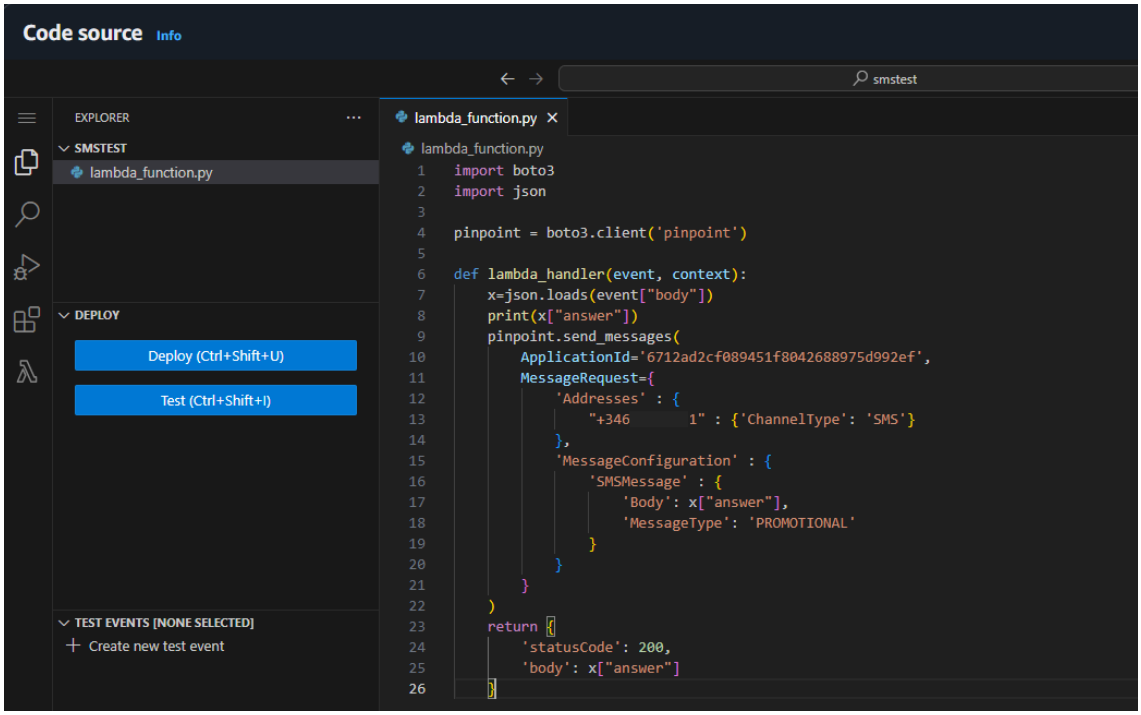
Crearemos una nueva función, en este caso vamos a desarrollarla en Python, por tanto, debemos elegir este lenguaje en los pasos de configuración.



A continuación, mostramos el código Python, el cual recibirá la petición a través de una petición HTTP encauzada por medio de un API Gateway, la procesa e invoca el servicio de Amazon Pinpoint.

En la sección “Code”, introducimos el código de nuestra función:

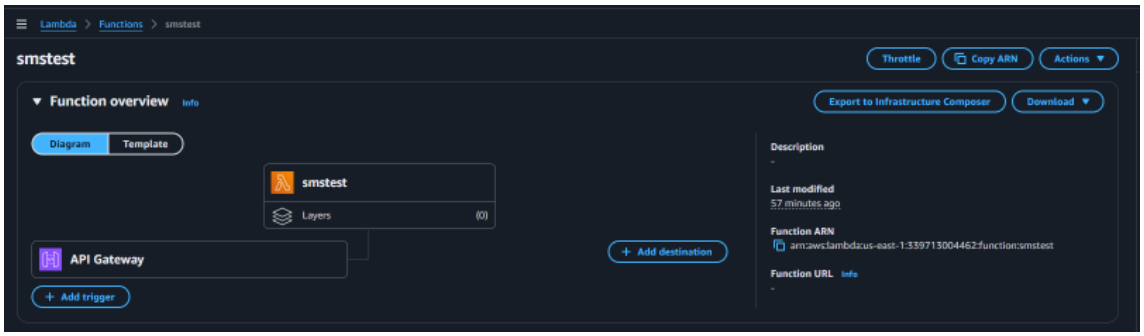
https://github.com/garrijuan/AWS-SMS-Notification/blob/main/pinpoint.py



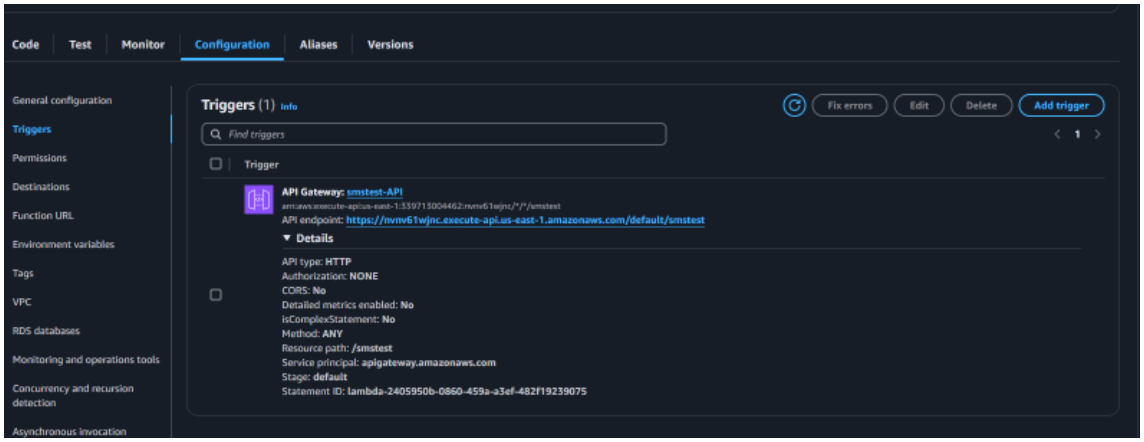
Hemos de indicar el “ProjectID” mencionado anteriormente para invocar la aplicación de mensajería. También hemos parametrizado el mensaje para enviar el deseado en cada momento, igualmente podríamos parametrizar el número de teléfono destino para enviar donde queramos.

Una vez que está listo el código, haremos “Deploy” de la función y tendremos la lambda operativa. En este caso para poder interactuar con lambda hemos creado un API Gateway para poder atacarla por medio de peticiones HTTP.

Para ello habría que añadir un Trigger en esta sección y configurarlo

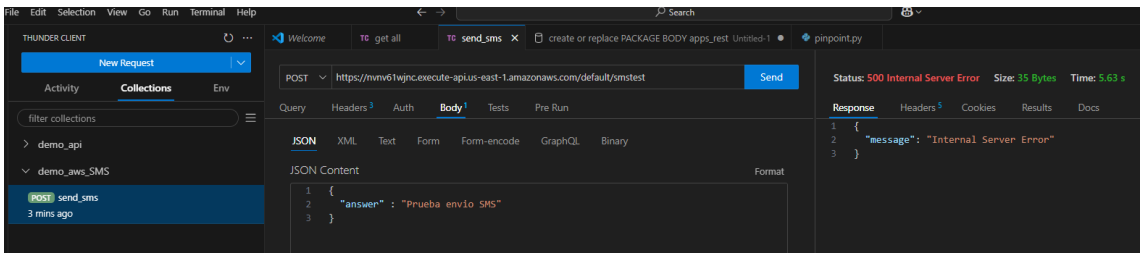


Endpoint:

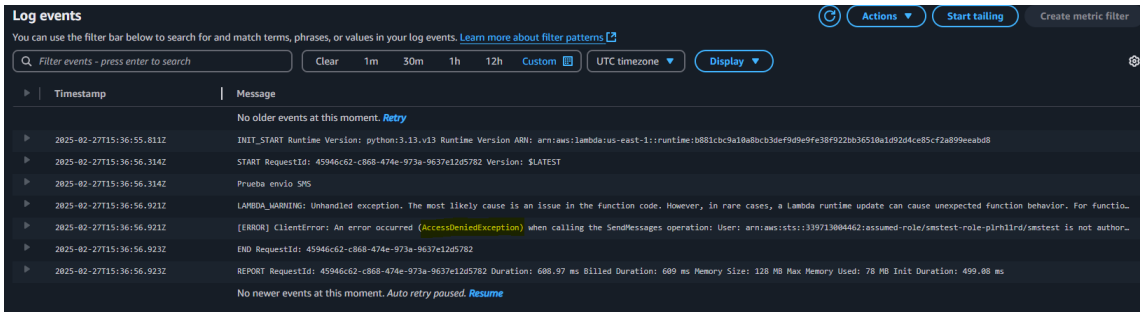


Tras todo esto, estamos preparados para una realizar una prueba “end to end”, preparamos un cliente para poder lanzar peticiones, en este caso usamos Thunder Client en VS Code.

A continuación, vemos que la petición ha fallado:



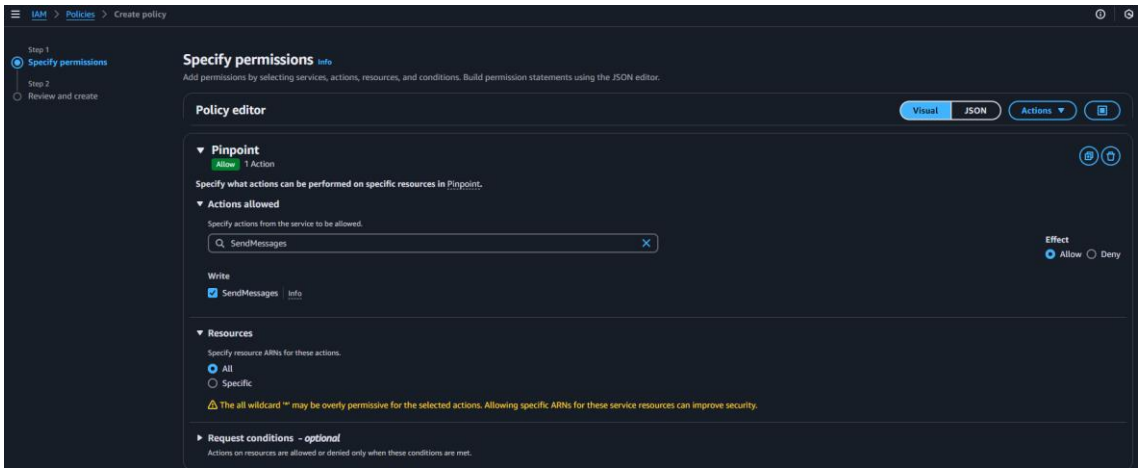
Log de lambda que muestra error de acceso denegado:



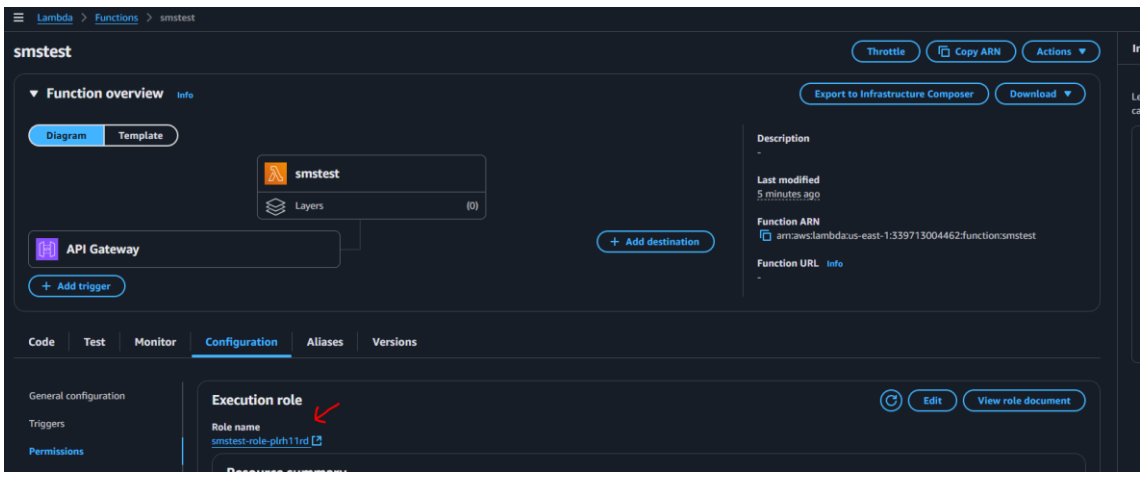
Policy

Es necesario crear una policy y asignarla al rol que creo automáticamente la lambda en su concepción.

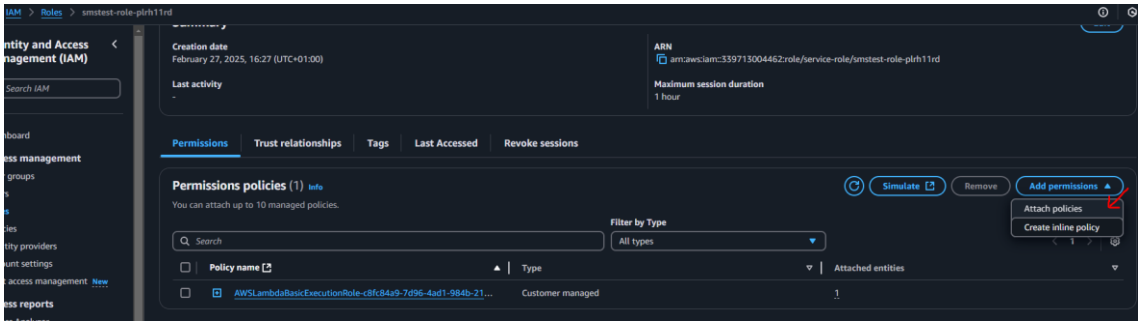
A continuación, creamos la policy:



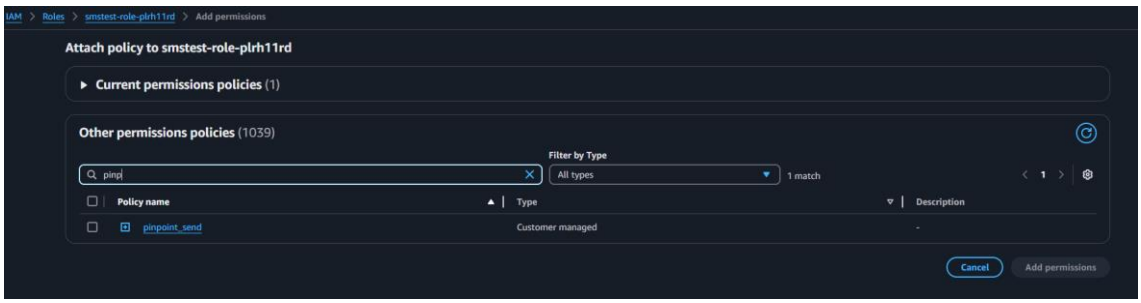
Asociación rol y policy:



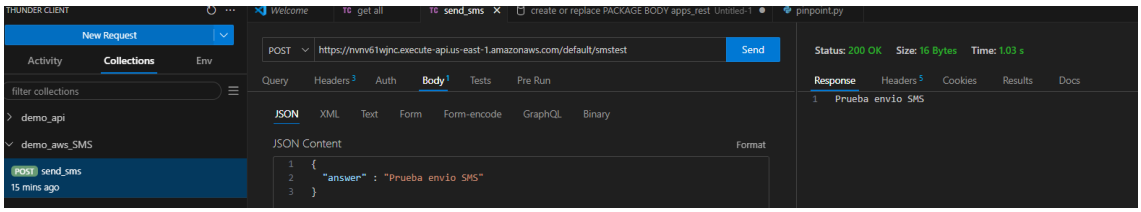
Clicar en attach policy:



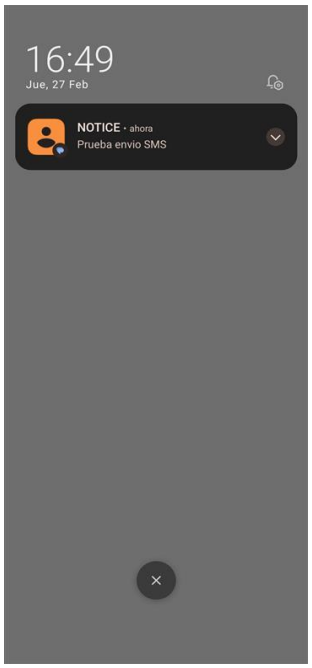
Elegir la policy creada anteriormente:



En este punto estamos preparados para repetir la prueba, desde Thunder Client lanzamos la petición:

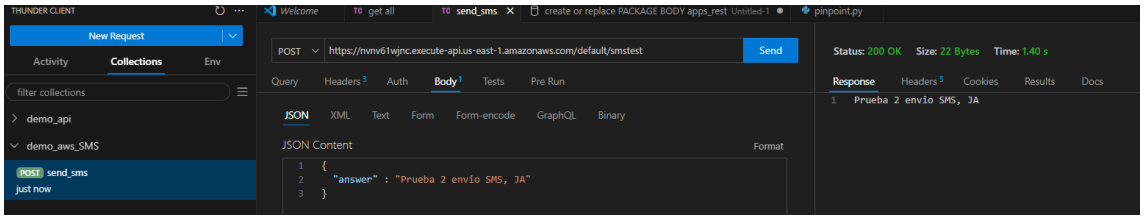


Ahora podemos observar que la petición se procesa correctamente.

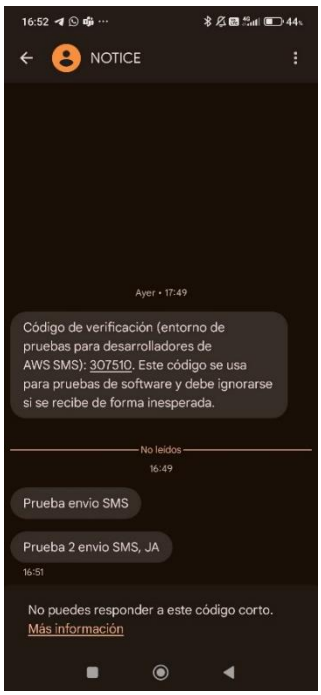


El envió del SMS se produce de forma inmediata.

Repetimos la prueba de nuevo proporcionando otro mensaje:



SMS recibido:



Conclusiones

En esta implementación, hemos utilizado Amazon API Gateway y AWS Lambda para crear un sistema eficiente y escalable para enviar notificaciones SMS a los clientes. Es importante tener en cuenta que tanto API Gateway como Lambda y Pintpoint, solo cobran por el uso, no por tenerlos desplegados y albergados en la nube, lo que permite un control de costos más efectivo.

La idea principal es que, desde cualquier aplicación o servicio invocador, puedan llamar a este Web Service para utilizar el sistema de comunicación, lo cual es extrapolable a otras nubes que ofrecen servicios similares. Por ejemplo, podríamos utilizar servicios como Twilio en lugar de Amazon Pinpoint para el envío de SMS, dependiendo de las necesidades y preferencias de la empresa.

Este enfoque no solo proporciona una solución serverless que reduce costos y simplifica la gestión, sino que también ofrece flexibilidad para adaptarse a diferentes proveedores y entornos en la nube, asegurando una comunicación eficiente y oportuna con los clientes.