



Certified Tech Developer

The Ultimate Degree

Infraestructura II

Actividad obligatoria e individual

Dificultad: Media

Monitoreo e Integraciones

Ya monitoreamos un RDS y una aplicación web. ¿Por qué no un poco de infraestructura?

Propuesta

En esta práctica proponemos monitorear el consumo de CPU de una máquina virtual EC2, agregar una alarma en CloudWatch que nos notifique en nuestro canal de Slack, y generar una carga para disparar la alarma.

Vamos a resolver esta ejercitación en las siguientes etapas:

- Levantar una máquina virtual EC2.
- Crear el alarmado en CloudWatch.
- Configurar la integra VM para disparar la alarma (podemos usar el comando stress de linux para esto).

¡Manos a la obra!

En la siguiente página se encuentra la resolución. Continúa únicamente para realizar una autoevaluación.

Resolución

- **Levantar una máquina virtual EC2**

Podemos levantar la VM de diversas maneras. En esta ocasión, vamos a hacerlo desde la consola, creando una instancia de tipo `t2.micro`. Vamos a cargar el siguiente userdata en el paso 5 de la sección “Detalles avanzados”. De esta manera, instalaremos nuestro comando para cargar la CPU.

```
#!/bin/bash

amazon-linux-extras install epel -y

yum install stress -y
```

▼ Detalles avanzados

Enclave ⓘ

☐ Habilitar

Metadatos accesibles ⓘ

Habilitado ▾

Versión de metadatos ⓘ

V1 y V2 (token opcional) ▾

Límite de saltos de respuesta de token de metadatos ⓘ

1 ▾

Datos de usuario ⓘ

☒ Como texto
 ☐ Como archivo
 ☐ La entrada ya está codificada en base64

```
usermod -a -G apache ec2-user
chown -R ec2-user:apache /var/www
chmod 2775 /var/www
find /var/www -type d -exec chmod 2775 {} \;
find /var/www -type f -exec chmod 0664 {} \;
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

Tipo ⓘ	Protocolo ⓘ	Rango de puertos ⓘ	Origen ⓘ
SSH ▾	TCP	22	Personaliza ▾ 0.0.0.0/0
Regla TCP per: ▾	TCP	80	Personaliza ▾ 0.0.0.0/0, ::/0
Regla TCP per: ▾	TCP	443	Personaliza ▾ 0.0.0.0/0, ::/0

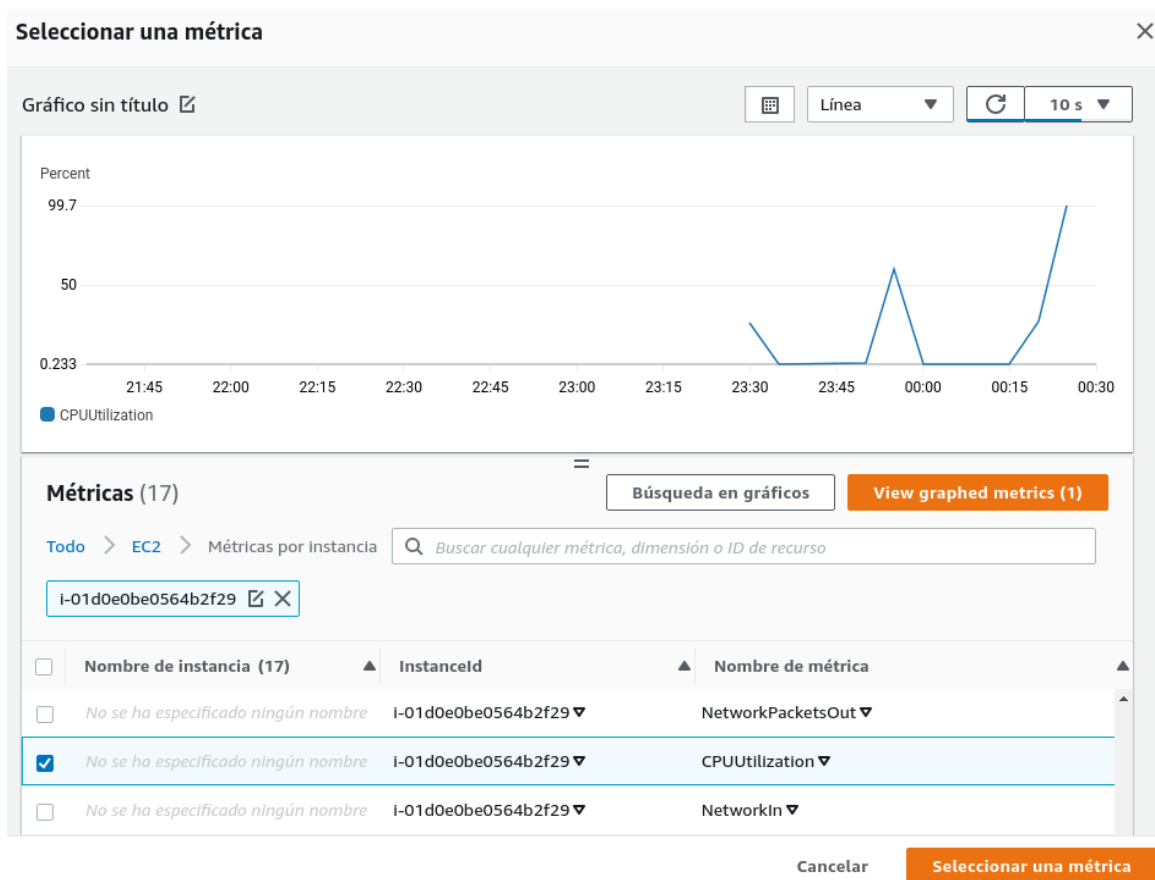
Para finalizar, descargamos la key y la creamos.

- **Crear el alarmado en CloudWatch**

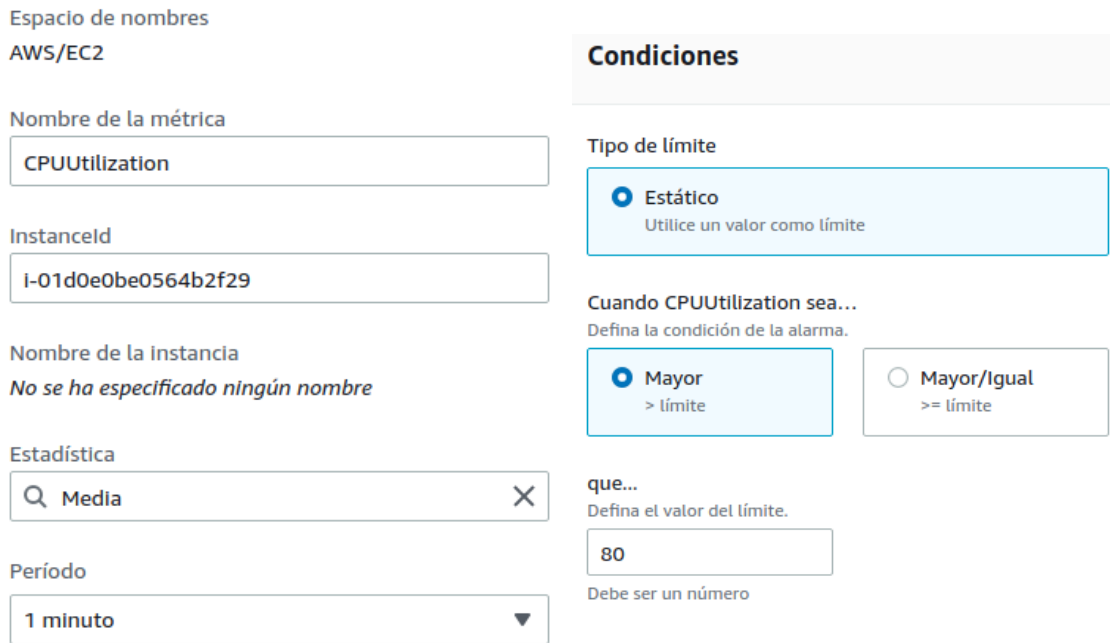
Para crear la alarma nos dirigimos al servicio de CloudWatch siguiendo el camino:

CloudWatch → Alarmas → Crear alarma → Seleccionar métrica

Elegimos EC2, "Métricas por instancia", filtramos por el ID de nuestra VM. Buscamos y seleccionamos "CPU Utilization" y presionamos el botón "Seleccionamos una métrica".



A continuación, establecemos los parámetros de la alarma tal y como se ve en las capturas:



The screenshot shows the AWS CloudWatch Alarm configuration interface. On the left, the 'Espacio de nombres' (Namespace) is set to 'AWS/EC2'. The 'Nombre de la métrica' (Metric Name) is 'CPUUtilization'. The 'InstanceId' is 'i-01d0e0be0564b2f29'. The 'Nombre de la instancia' (Instance Name) is empty, with a note 'No se ha especificado ningún nombre'. The 'Estadística' (Statistic) is 'Media' (Average). The 'Período' (Period) is '1 minuto'. On the right, the 'Condiciones' (Conditions) section is active. The 'Tipo de límite' (Limit Type) is 'Estático' (Static). The 'Cuando CPUUtilization sea...' (When CPUUtilization is...) section shows two options: 'Mayor > límite' (Greater than limit) and 'Mayor/Igual >= límite' (Greater than or equal to limit). The 'que...' (that...) section shows the limit value '80'.

Espacio de nombres
AWS/EC2

Nombre de la métrica
CPUUtilization

InstanceId
i-01d0e0be0564b2f29

Nombre de la instancia
No se ha especificado ningún nombre

Estadística
Media

Período
1 minuto

Condiciones

Tipo de límite
☒ Estático
Utilice un valor como límite

Cuando CPUUtilization sea...
Defina la condición de la alarma.

☒ Mayor
> límite

☐ Mayor/Igual
>= límite

que...
Defina el valor del límite.

80

Debe ser un número

Hacemos clic en siguiente.

Configurados estos parámetros, el paso que sigue es marcar una acción en caso de alarma. En primera instancia, definimos que la notificación será a través del correo electrónico. Seleccionamos:

- En modo Alarma.
- Crear un nuevo tema. Debajo colocamos las direcciones de correo electrónico y hacemos clic en "Crear tema". **IMPORTANTE:** recordar el nombre de nuestro tema, lo vamos a necesitar más adelante.

Configurar las acciones

Notificación

Activador de estado de alarma

Definir el estado de alarma que activará esta acción.

Eliminar



En modo alarma

La métrica o expresión se encuentra fuera del límite definido.



CORRECTO

La métrica o expresión está dentro del límite definido.



Datos insuficientes

La alarma se acaba de iniciar o no hay suficientes datos disponibles.

Seleccione un tema de SNS

Defina el tema de SNS (Simple Notification Service) que recibirá la notificación.

☐ Seleccione un tema de SNS existente

☒ Crear un tema nuevo

☐ Usar el ARN del tema

Crear un nuevo tema...

El nombre del tema debe ser único.

TemaNotificacionApp

Los nombres de los temas de SNS solo pueden contener caracteres alfanuméricos, guiones (-) y guiones bajos (_).

Puntos de enlace de correo electrónico que recibirán la notificación...

Añada una lista de direcciones de correo electrónico separadas por comas. Cada dirección se agregará como una suscripción al tema anterior.

deck2k@gmail.com

usuario1@ejemplo.com, usuario2@ejemplo.com

Siguiente paso: asignamos un nombre a la alarma. Quedará guardada en nuestro panel de alarmas.

Se ha creado correctamente la alarma **Alarma4xxApp**. [Ver alarma](#)

Algunas suscripciones están pendientes de confirmación
Amazon SNS no envía mensajes a un punto de conexión hasta que se confirma la suscripción [Ver suscripciones a SNS](#)

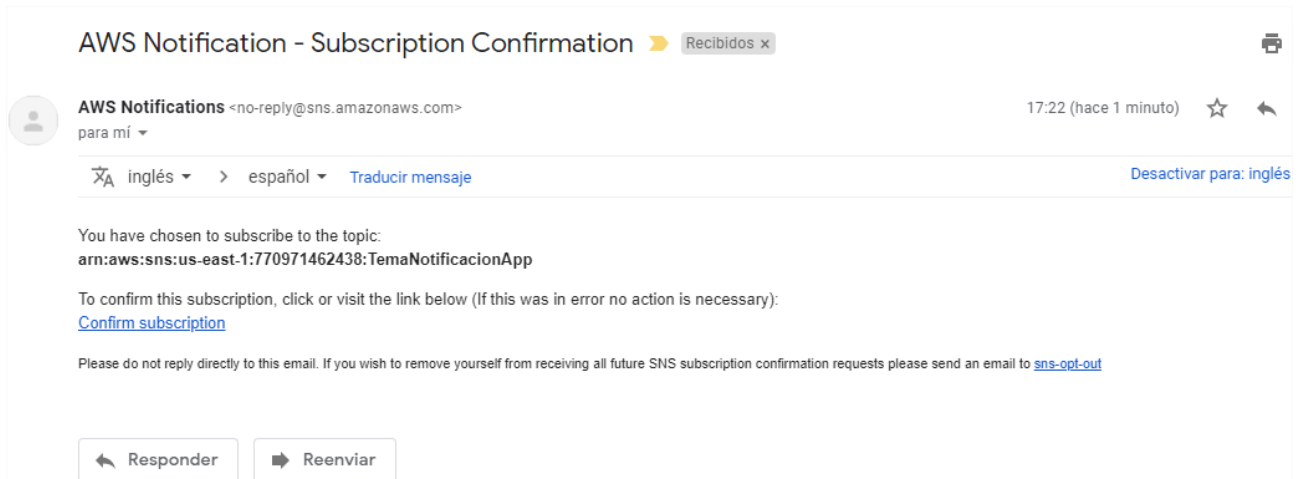
CloudWatch > Alarmas

Alarmas (1) ☐ Ocultar alarmas de Auto Scaling [Borrar selección](#) [Crear alarma compuesta](#) [Acciones](#) [Crear alarma](#)

[Cualquier es...](#) [Cualquier tipo](#) < 1 > [Configurar](#)

<input type="checkbox"/>	Nombre	Estado	Última actualización del estado	Condiciones	Acciones
<input type="checkbox"/>	Alarma4xxApp	Datos insuficientes	2021-09-13 17:23:21	ApplicationRequests4xx > 5 para 1 puntos de datos dentro de 1 minuto	Acciones habilitadas Advertencia

Nota importante: cuando creamos el tema y cargamos allí los correos electrónicos, llegará un mensaje a cada uno de ellos en donde deben confirmar la suscripción a este canal de alarmas. El correo que se envía es similar a este:



Una vez suscriptos, recibiremos las notificaciones de esta alarma.

- **Configurar la integración para notificar a Slack.**

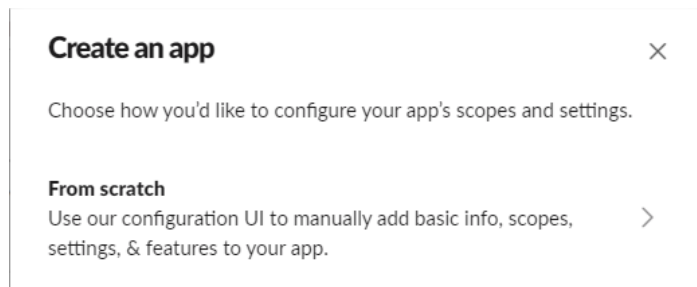
» Creación de un Webhook en Slack

¡Ya podemos ser notificados a través del correo electrónico! Pero **AWS CloudWatch nos puede notificar mediante otros canales** como Notificaciones Push hacia aplicaciones móviles creadas por nosotros, SMS, o –como este caso– mensajes en **un canal de Slack**.

Para hacer esta configuración, nos logueamos a Slack en su versión web y vamos a este link: <https://api.slack.com/messaging/webhooks>. Allí debemos realizar los siguientes pasos:

- Crear una aplicación en Slack (<https://api.slack.com/apps/new>), en modalidad “From scratch”:

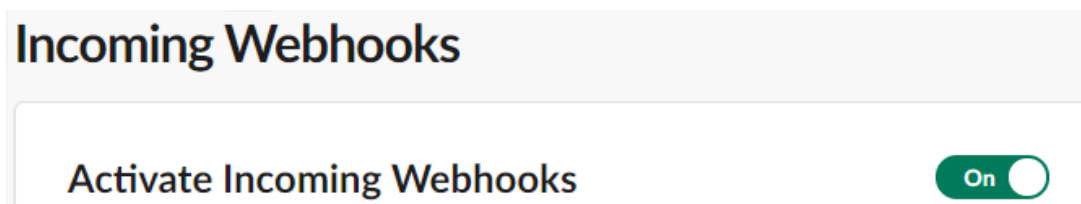




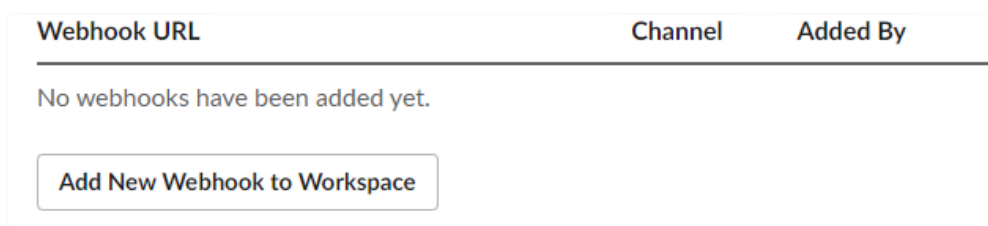
Indicamos un nombre y a qué Workspace vamos a querer notificar. Una vez creada la app, nos dirigimos al menú de opciones de nuestra aplicación en Slack: <https://api.slack.com/apps>

Aquí debemos **habilitar el Webhook**. Esto es, básicamente, exponer un método HTTPS que permite que otra aplicación o código pueda notificar en dicho canal. Vamos a:


Add features and functionality → Incoming Webhooks → Lo activamos



Vamos luego hasta el final de la página y agregamos un nuevo Webhook, con la opción **Add New Webhook to Workspace:**



Nos solicitará en qué canal vamos a autorizar a esta app a publicar los mensajes (podemos crear tantos Webhooks como queramos). Una vez creado, vamos a ver lo siguiente:

Webhook URL	Channel	Added By
<input type="text" value="https://hooks.slack.com/services/T02D"/> <input type="button" value="Copy"/>	#general	deck2k Sep 13, 2021 
<input type="button" value="Add New Webhook to Workspace"/>		

Notamos que tenemos un **Webhook URL**, ¡lo copiamos!

» Integración en AWS Lambda

Ya generado el Webhook en Slack, necesitamos invocarlo desde el ecosistema AWS. Para ello vamos a utilizar uno de sus servicios englobados dentro los denominados "Serverless": **AWS Lambda**. Este servicio nos va a permitir agregar una lógica personalizada a nuestras notificaciones de CloudWatch.

Nos dirigimos al Servicio AWS Lambda y allí vamos a "Crear nueva función" y seleccionamos las opciones:

- Crear desde cero
- Tiempo de ejecución: Python 3.9

Seleccione una de las siguientes opciones para crear la función.

Crear desde cero ☒

Empiece con un sencillo ejemplo "Hello World".

Utilizar un proyecto ☐

Cree una aplicación Lambda utilizando un código de muestra y los ajustes de configuración predefinidos de casos de uso comunes.

Imagen del contenedor ☐

Seleccione una imagen de contenedor para implementar para la función.

Información básica

Nombre de la función
Escriba un nombre para describir el propósito de la función.

Utilice exclusivamente letras, números, guiones o guiones bajos. No incluya espacios.

Tiempo de ejecución [Info](#)
Elija el lenguaje que desea utilizar para escribir la función. Tenga en cuenta que el editor de código de la consola solo admite Node.js, Python y Ruby.

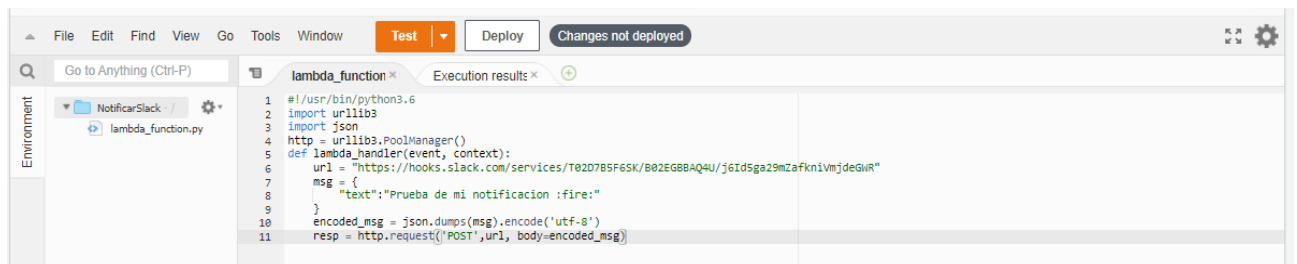
Permisos [Info](#)
De forma predeterminada, Lambda creará un rol de ejecución con permisos para cargar registros en Amazon CloudWatch Logs. Puede personalizar este rol predeterminado más adelante.

► [Cambiar el rol de ejecución predeterminado](#)

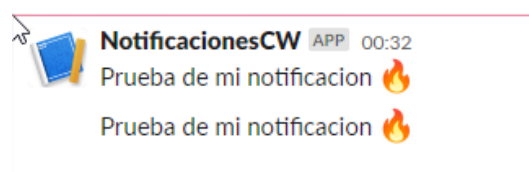
Una vez allí, reemplazamos el código que aparece en **lambda_function.py** por el siguiente:

```
#!/usr/bin/python3.6
import urllib3
import json
http = urllib3.PoolManager()
def lambda_handler(event, context):
    url = "URL DEL WEBHOOK DE SLACK"
    msg = {
        "text": "Prueba de mi notificacion :fire:"
    }
    encoded_msg = json.dumps(msg).encode('utf-8')
    resp = http.request('POST', url, body=encoded_msg)
```

Solo reemplazamos el valor de URL por la URL del Webhook de Slack del paso anterior:

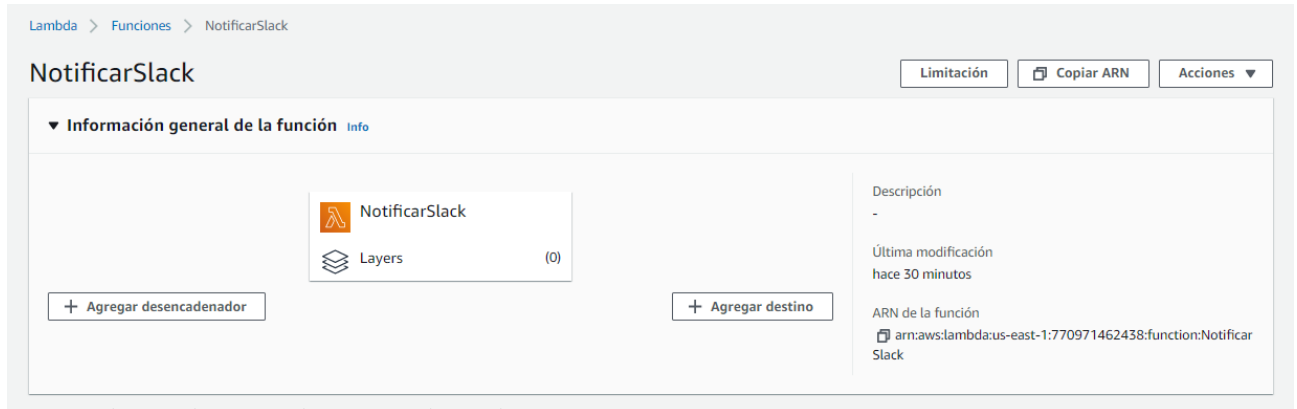


Una vez realizado ese paso, debemos *deployar* la función. Luego, podemos probarla haciendo clic en Test. **Si nuestra integración fue exitosa, ¡vamos a recibir el mensaje en el canal de Slack!**

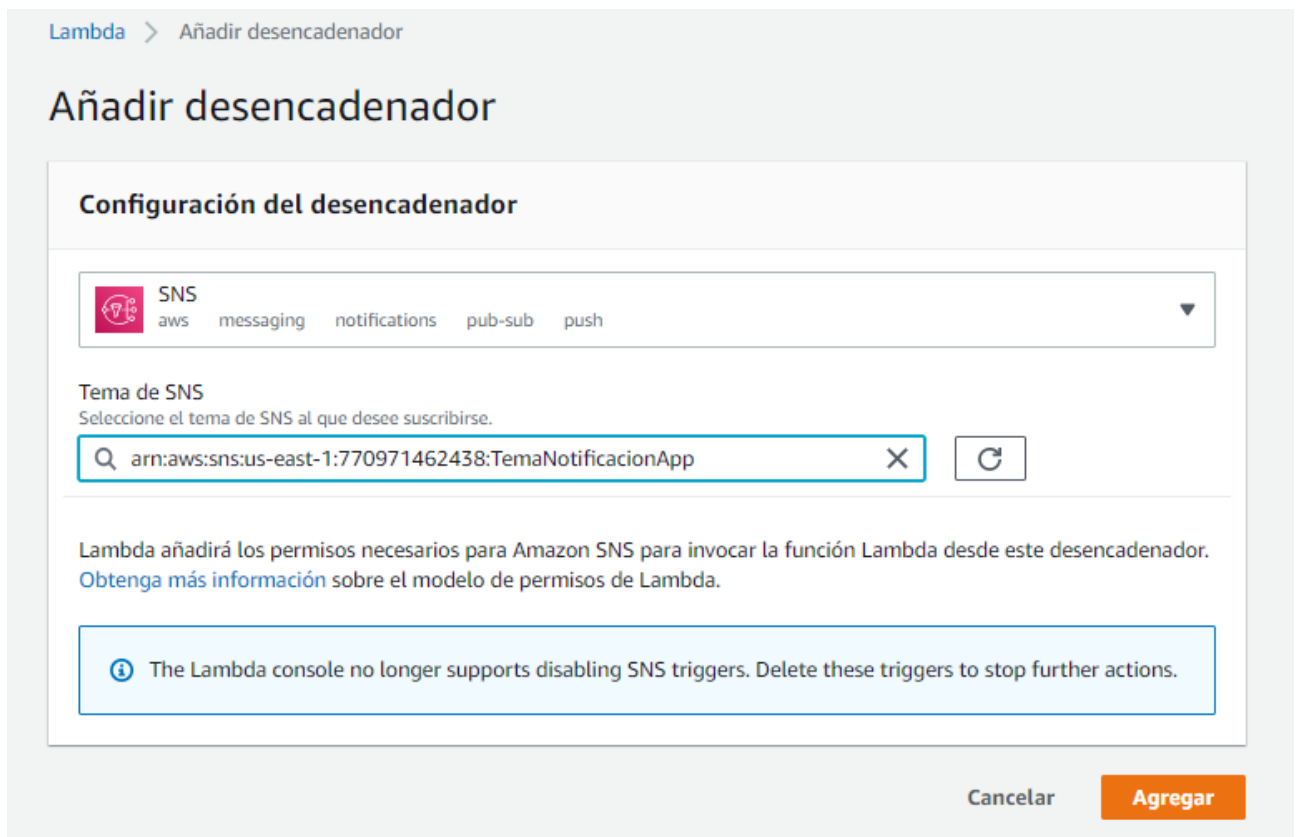


El último paso consiste en integrar esta función con el desencadenante de CloudWatch. Para ello, vamos a asociarla con el tema creado en las notificaciones de CloudWatch.

En el menú principal de nuestra función Lambda, vamos a “**Agregar desencadenador**”:



En la selección de servicios, buscamos SNS y lo seleccionamos. Luego elegimos el tema correspondiente:



¡Ya tenemos la alarma lista!

- **Generar carga a la VM para disparar la alarma.**

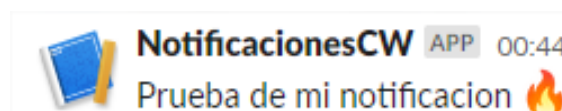
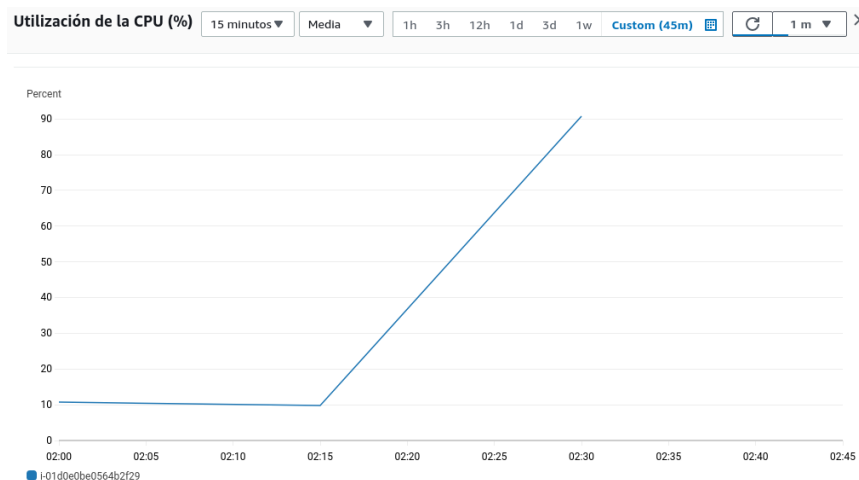
Ingresamos ahora a nuestra máquina virtual para generar carga y elevar el consumo de la CPU. Para iniciar, utilizamos un comando como el siguiente con nuestro key.pem que descargamos al crear la máquina y la IP pública de la misma.

```
$ ssh -i key.pem ec2-user@54.160.247.89
```

Para generar la carga utilizaremos el comando stress instalado con el userdata al instanciar la VM, de la siguiente forma:

```
$ # stress --cpu 1 --vm 1 --vm-bytes 128M
```

Podremos notar en la sección de monitoreo de nuestra instancia como aumenta el uso del CPU, lo que producirá que se active nuestra alarma y nos notifique en Slack.



Notificación de Slack

Conclusión

En esta práctica vimos que no solo podemos alertar y notificar alertas de aplicaciones sino **también de infraestructura**. Es importante establecer los monitoreos y alarmas adecuados, ya que podemos tener una falla en una aplicación debida a una falla en infraestructura.

¡Controlar cada aspecto generando notificaciones –como en los casos indicados– nos permitirá encontrar más rápido la causa del problema!