



## Infraestrutura II

**Atividade obrigatória e individual**

**Dificuldade: média**

# Monitoramento e integrações

Neste ponto, estamos claros que o controle dependerá da observabilidade com a qual configuramos nossos serviços de monitoramento. Seria inútil implantar agentes se não definíssemos políticas de alerta integradas aos nossos sistemas de e-mail ou pagers para quando um problema detectado não puder ser corrigido automaticamente.

Porém, a integração não é exclusiva do sistema de alerta! No próximo exercício, veremos que incorporar o monitoramento a outras ferramentas nos dará maior valor agregado e mais produtividade.

## Do que se trata?

Vamos conectar os serviços AWS existentes com os dados produzidos pelo sistema de monitoramento. As integrações nos dão a capacidade de serviços e aplicativos comunicarem informações em tempo real e nos permitem acionar ações para usuários e aplicativos.

Vejamos três qualidades que emergem do uso dessa habilidade:

- Notificações em tempo real.
- Integração e processamento de dados.
- Eventos de segurança.

**Vamos examinar cada um deles!**

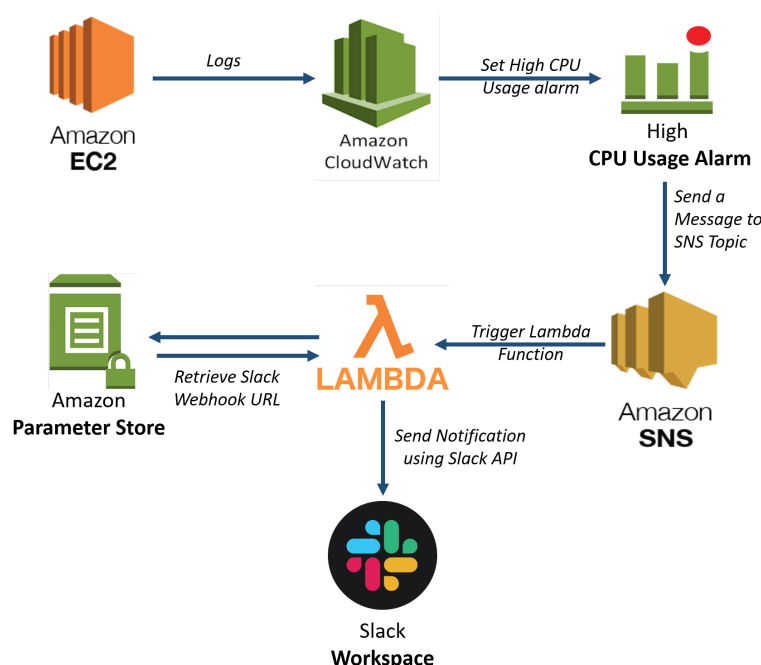
## Notificações em tempo real

Hoje, os princípios just-in-time levaram a uma migração em grande escala das operações tradicionais. O uso de e-mail como modelo de notificação ou de processos em lote, que exigia algum tempo de processamento para despachar uma única mensagem, foi substituído por outras opções.

O padrão para muitos negócios em que a velocidade das notificações é primordial, baseia-se na detecção e reação imediata a eventos gerados em outra parte do sistema.

Por exemplo, podemos encadear CloudWatch, Parameter Store, Lambda e Slack para serem notificados em tempo real por meio de um pop-up em um canal Slack, quando o status de uma instância EC2 muda para um valor definido como crítico.

Isso servirá para ser alertado imediatamente e não dependerá de uma notificação assíncrona, como e-mail.



*Gerando notificações para o Slack*

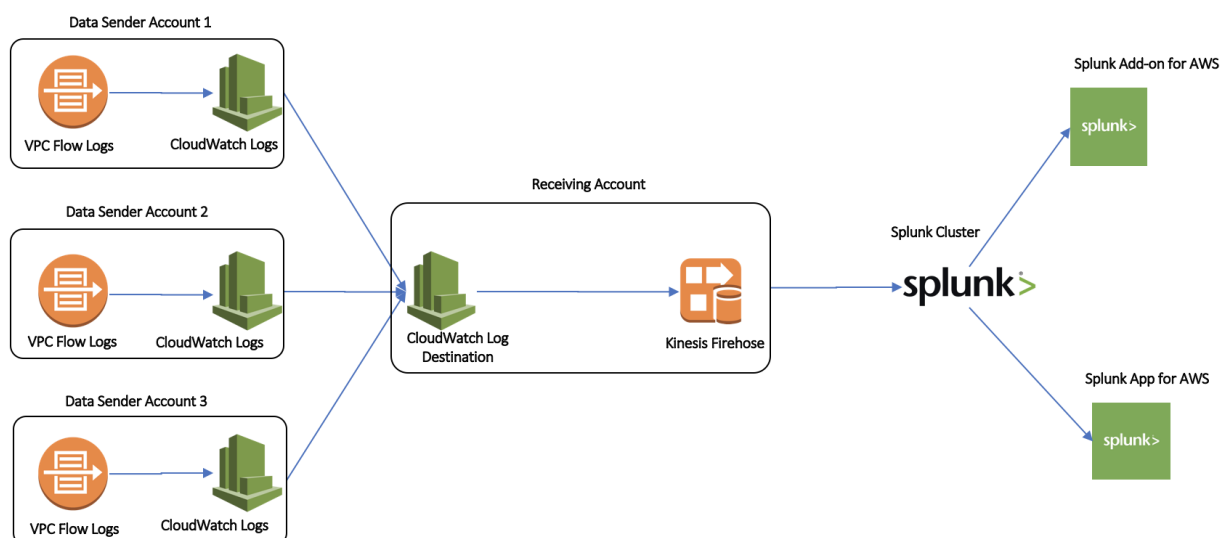
## Integração e processamento de dados

A integração e o processamento de dados é o processo de combinar dados de várias fontes em uma visão unificada. O objetivo, em termos gerais, será obter informações significativas e inteligência acionável.

Dependendo do nosso negócio, os dados podem crescer exponencialmente em volume, vir em diferentes formatos e ser distribuídos em diferentes serviços em nosso ambiente.

Diante desse cenário, as ferramentas de integração de dados terão como objetivo agregar dados independentemente de seu tipo, estrutura ou volume.

No exemplo a seguir, vemos os serviços de Flow Logs enviarem dados para o CloudWatch e, a partir daí, enviá-los para um coletor central que será ingerido pelo Kinesis e processado. Finalmente, eles serão enviados ao Splunk para serem exibidos de uma forma significativa para nós.

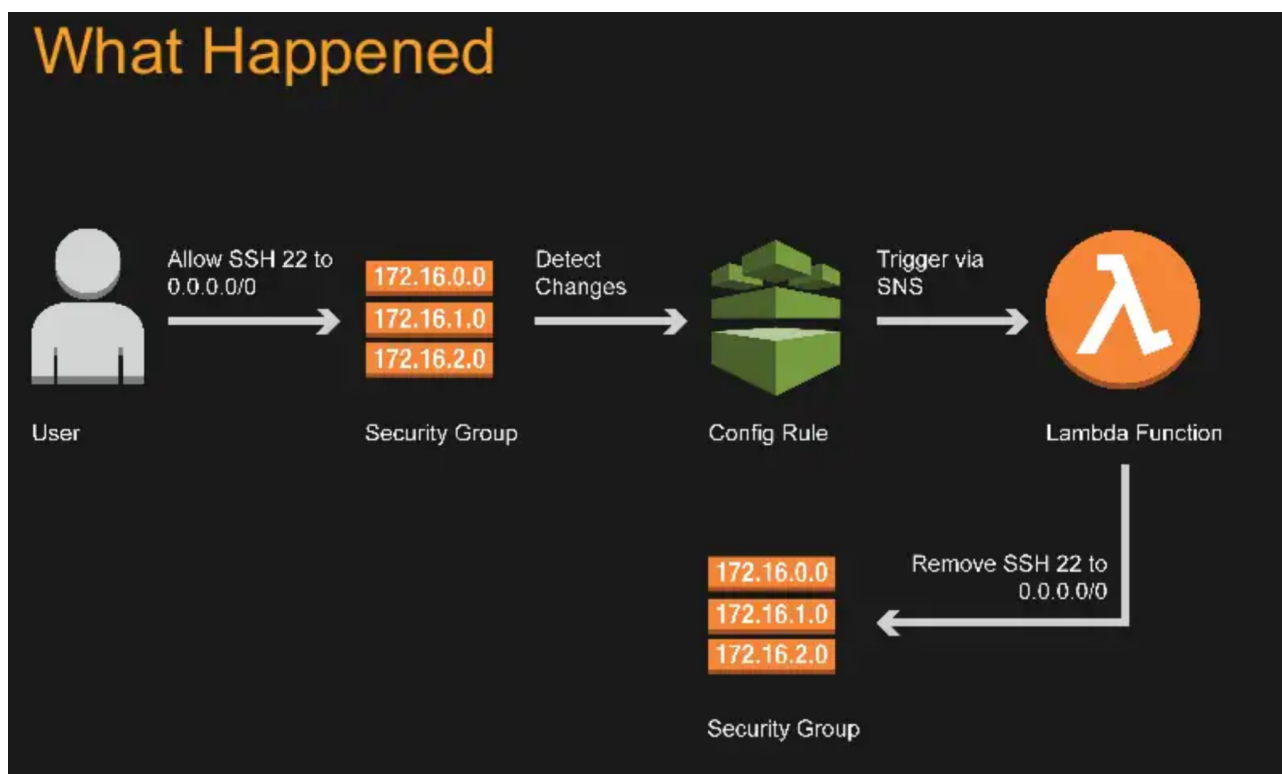


*Fluxo de dados para armazenamento e processamento*

## Eventos de segurança

Por ter um sistema de resposta orientado a eventos, teremos um mecanismo ativo de detecção e resposta para remediar o evento automaticamente. Este sistema é usado atualmente por vários produtos na indústria de monitoramento de segurança.

Um exemplo com a AWS seria um usuário fazendo alterações não autorizadas em grupos de segurança. Essas alterações serão refletidas nas regras de configuração e enviadas para uma função Lambda que retornará as alterações ao estado original.



*Acionamento com base em um evento*

## Mãos a obra

Em nosso cenário, iremos **notificar um canal Slack da ativação de um alarme.**

Vamos dividir a atividade em quatro fases:

1. Criação do APP no AWS Elastic Beanstalk.

2. Criação de alarmes no AWS CloudWatch.
3. Criação do Webhook no Slack.
4. Integração no AWS Lambda.

## 1. Criação do APP em Elastic Beanstalk.

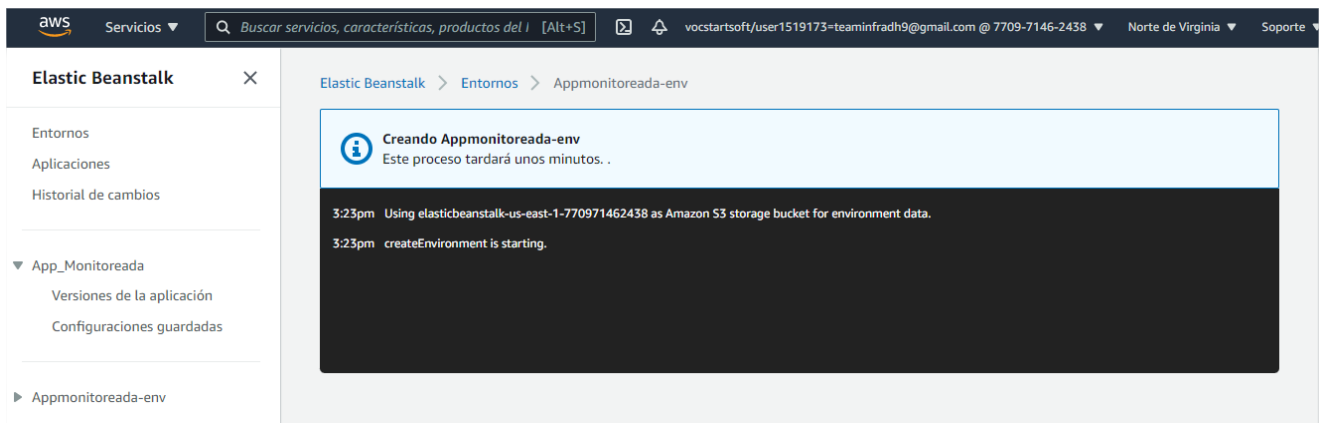
Uma vez conectado ao AWS Educate e em nosso console da Web, vamos para o serviço Elastic Beanstalk e lá vamos para "Criar aplicativo":



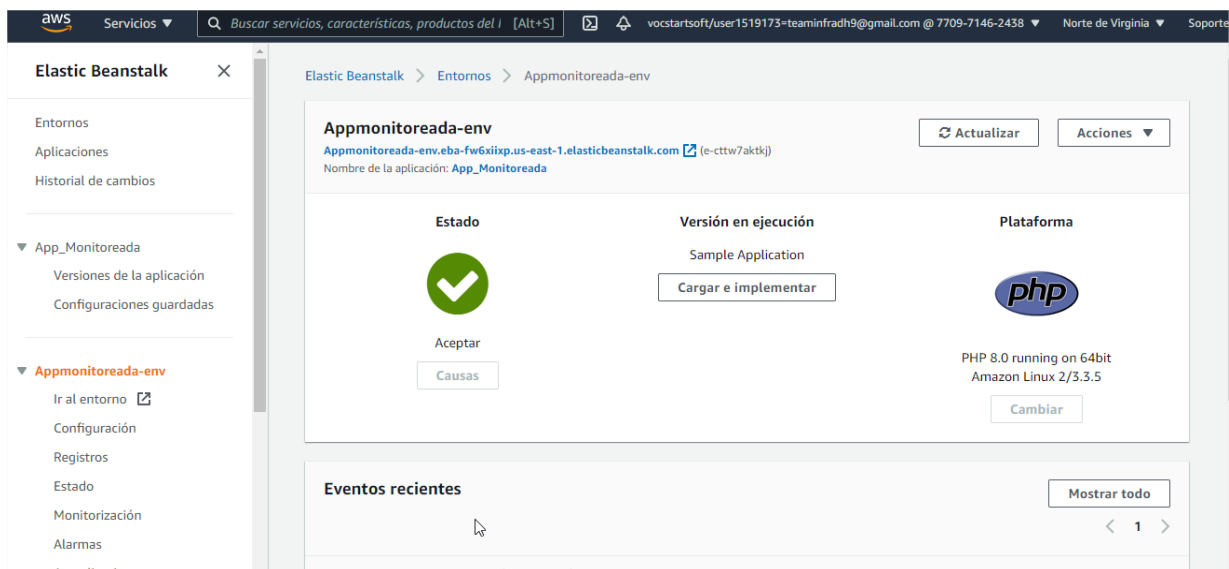
Devemos atribuir alguns parâmetros a ele:

- Nome do aplicativo: **App\_Monitorado**
- Plataforma: **PHP**
- Código do aplicativo: **aplicativo de amostra**

O serviço em primeira instância implantará os recursos necessários para a aplicação e os gerará em uma estrutura, chamada de ambiente, processo que pode levar até 10 minutos.



Após este processo, podemos ver o status do aplicativo e a URL de acesso.



A próxima etapa é expor as métricas para poder monitorar o aplicativo. Em nosso caso, vamos expor aqueles que contam erros HTTP 4xx (404, 401, etc.).

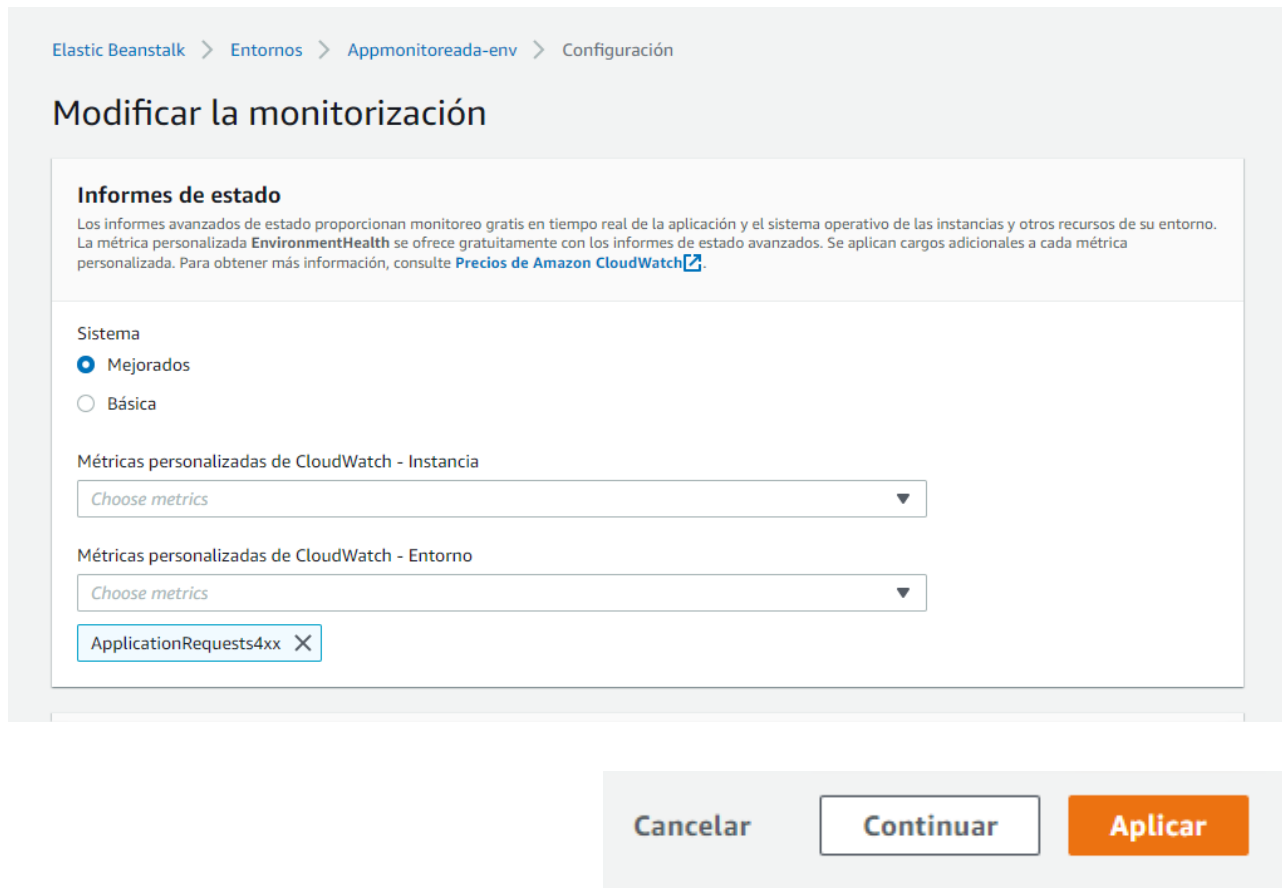
Vamos para a seção de nosso ambiente:

**Configuração → Monitoramento → Editar**

E lá nós selecionamos - dentro do combo "Custom Environment CloudWatch Metrics" -:

- **ApplicationRequest4xx**

**E aplicamos as alterações (na seção inferior direita, botão "Aplicar"):**



Elastic Beanstalk > Entornos > Appmonitoreada-env > Configuración

### Modificar la monitorización

**Informes de estado**

Los informes avanzados de estado proporcionan monitoreo gratis en tiempo real de la aplicación y el sistema operativo de las instancias y otros recursos de su entorno. La métrica personalizada **EnvironmentHealth** se ofrece gratuitamente con los informes de estado avanzados. Se aplican cargos adicionales a cada métrica personalizada. Para obtener más información, consulte [Precios de Amazon CloudWatch](#).

Sistema

☒ Mejorados

☐ Básica

Métricas personalizadas de CloudWatch - Instancia

Choose metrics ▼

Métricas personalizadas de CloudWatch - Entorno

Choose metrics ▼

ApplicationRequests4xx X

Cancelar Continuar Aplicar

Essas mudanças reiniciam nosso ambiente. Lembre-se de que o aplicativo ficará indisponível por alguns instantes.

## 2. Criação de alarmes em AWS CloudWatch

Para iniciar esta segunda fase, fomos ao CloudWatch com o objetivo de gerar alarmes com base nas métricas expostas.

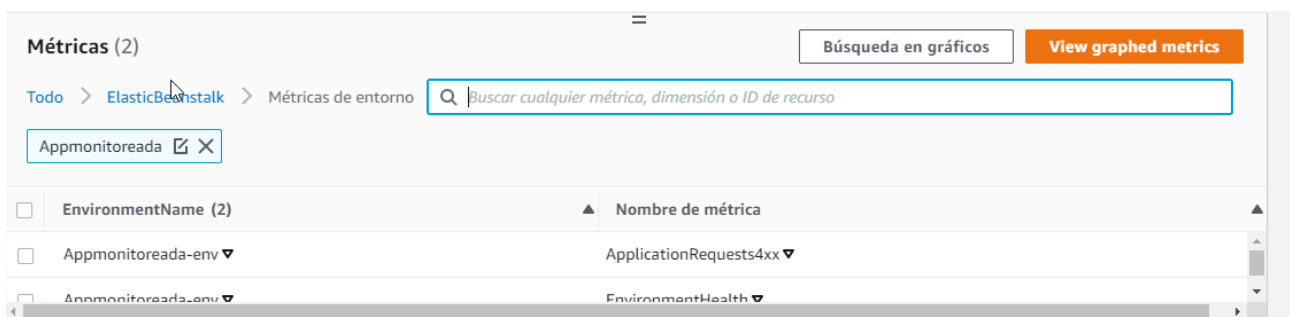
Uma vez dentro do serviço, vamos criar um alarme:

**CloudWatch → Alarmes → Criar alarme → Selecione a métrica**

Veremos lá todas as métricas disponíveis, classificadas de acordo com cada serviço AWS (EC2, RDS, Elastic Beanstalk, etc.):



Selecionamos Elastic Beanstalk e, lá, "Environment Metrics". Em seguida, filtramos a pesquisa pelo nome de nosso aplicativo e ele listará suas métricas (aparece ApplicationRequest4xx):



Com a métrica selecionada, devemos configurar o que queremos medir e quais são os limites de alarme. Em nosso caso, vamos medir o seguinte:

- Se houver mais de cinco solicitações que retornaram um status HTTP 4xx no último minuto, o alarme será disparado.



### Métrica

Gráfico

Esta alarma se activará cuando la línea azul vaya encima la línea roja por 1 puntos de datos dentro de 1 minuto.

Count

5

4

3

2

1

0

18:00 19:00 20:00

ApplicationRequests4xx

Editar

Espacio de nombres  
AWS/ElasticBeanstalk

Nombre de la métrica  
ApplicationRequests4xx

EnvironmentName  
Appmonitoreada-env

Estadística  
Media

Período  
1 minuto

### Condiciones

Tipo de límite

☒ Estático  
Utilice un valor como límite

☐ Detección de anomalías  
Utilice una banda como límite

Cuando ApplicationRequests4xx sea...

Defina la condición de la alarma.

☒ Mayor  
> límite

☐ Mayor/Igual  
>= límite

☐ Menor/Igual  
<= límite

☐ Menor  
< límite

que...

Defina el valor del límite.

5

Debe ser un número

Configuración adicional

Cancelar **Siguiente**

Uma vez configurados esses parâmetros, o seguinte é marcar uma ação em caso de alarme. Em primeira instância, definimos que a notificação será por e-mail.

Seleccionamos:

- Em modo de alarme.
- Crie um novo tópico, abaixo colocamos os endereços de e-mail e clicamos em "Criar tópico". **IMPORTANTE:** lembre-se do nome do nosso tema, vamos precisar dele mais tarde.

### Configurar las acciones

#### Notificación

Activador de estado de alarma

Definir el estado de alarma que activará esta acción.

☒ 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.

Eliminar

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 (\_).

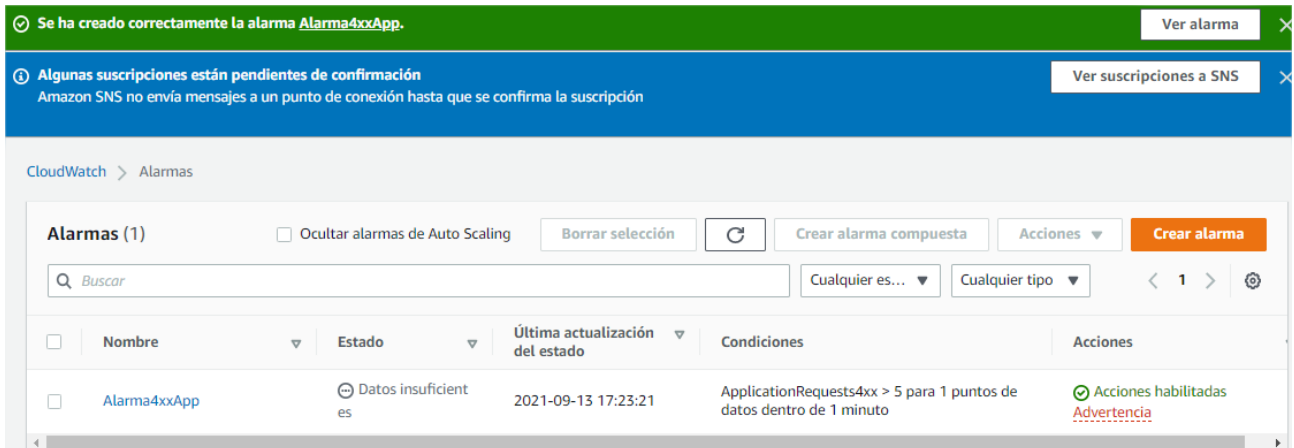
Puntos de enlace de correo electrónico que recibirán la notificación...

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

Vamos para a próxima etapa: atribuímos um nome ao alarme e ele será salvo em nosso painel de alarme.



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](#)

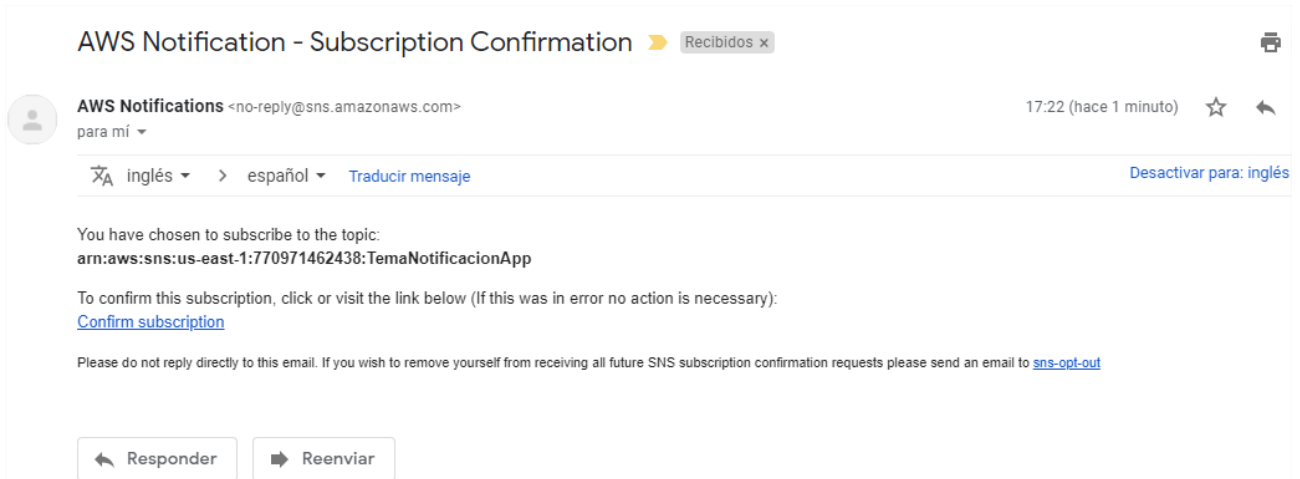
CloudWatch > Alarmas

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

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

<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	<a href="#">Acciones habilitadas</a> <a href="#">Advertencia</a>

**NOTA IMPORTANTE:** Quando criamos o tópico e carregamos os emails lá, uma mensagem chegará a cada um deles onde eles devem confirmar a inscrição neste canal de alarme. O e-mail enviado é semelhante a este:



**AWS Notification - Subscription Confirmation** [Recibidos](#)

**AWS Notifications** <no-reply@sns.amazonaws.com> 17:22 (hace 1 minuto) [★](#) [↶](#)  
para mí [▼](#)

[🌐](#) [Inglés](#) > [español](#) [Traducir mensaje](#) [Desactivar para: inglés](#)

You have chosen to subscribe to the topic:  
**arn:aws:sns:us-east-1:770971462438:TemaNotificacionApp**

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):  
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

[↶ Responder](#) [➡ Reenviar](#)

Uma vez inscrito, podemos receber notificações deste alarme.

**Você tem coragem de gerar este alarme?**

**Como você poderia gerar solicitações que retornam algum erro 4xx para acioná-lo?**

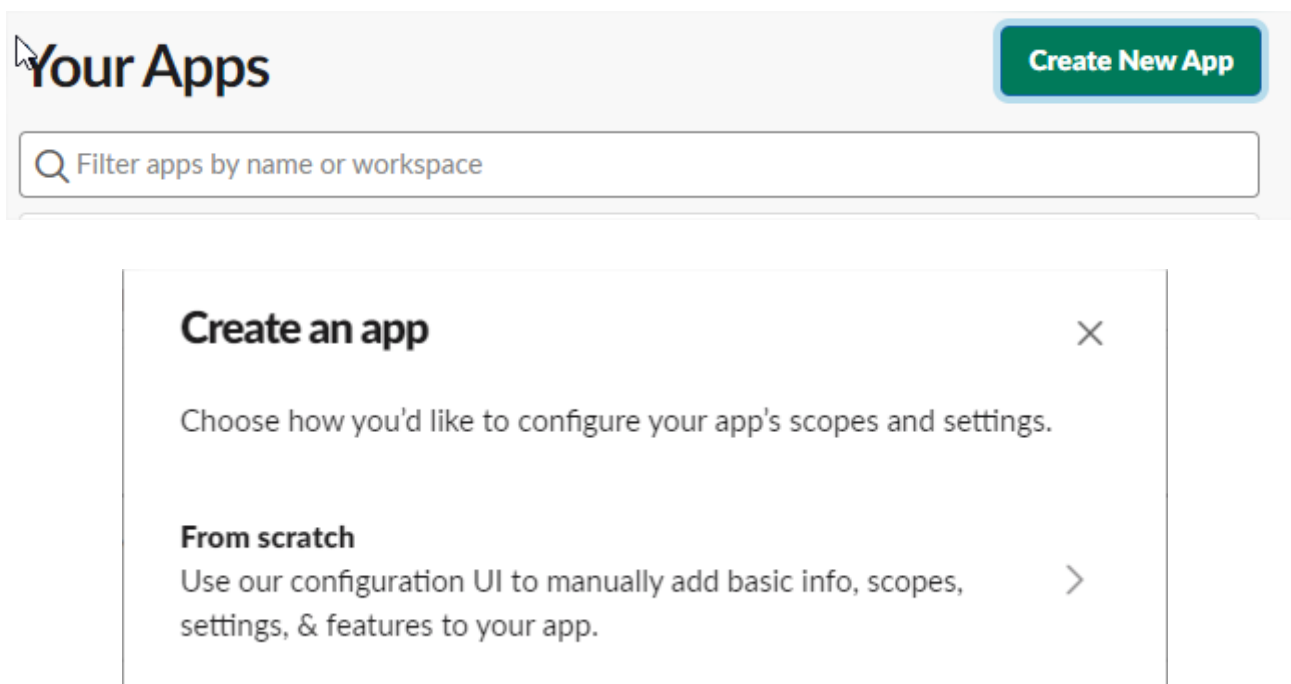
### 3. Crie um webhook no Slack

Com certeza você já pode receber os alarmes no e-mail. Mas o AWS **CloudWatch pode nos notificar por meio de outros canais**, como notificações push para aplicativos móveis criados por nós, SMS ou - como neste caso - mensagens em um canal Slack.

Para fazer essa configuração, logamos no Slack em sua versão Web e vamos a este link: <https://api.slack.com/messaging/webhooks>

Lá devemos realizar as seguintes etapas:

- Crie um aplicativo no Slack (<https://api.slack.com/apps/new>), no modo "Do zero":



Lá indicamos um nome e qual espaço de trabalho queremos notificar.

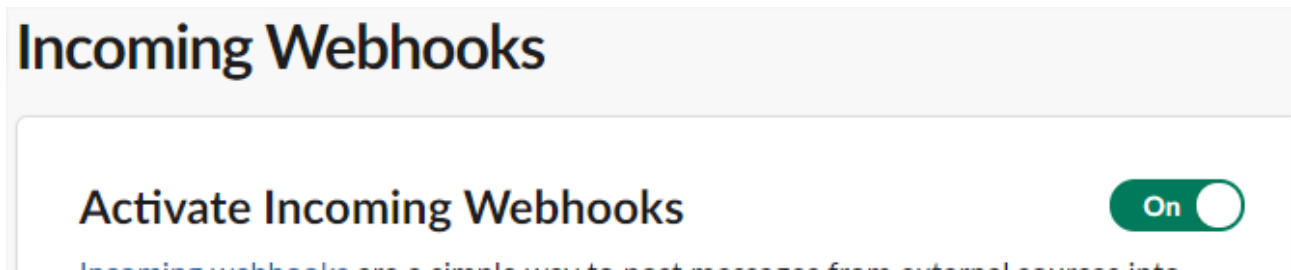
Depois de criado, vamos para o menu de opções de nosso aplicativo no Slack:

<https://api.slack.com/apps>

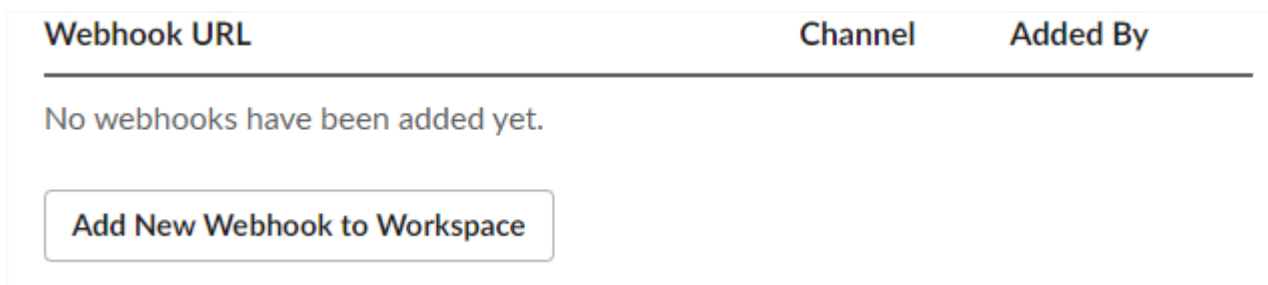
Aqui, devemos **habilitar o webhook**. Isso basicamente expõe um método HTTPS que permite que outro aplicativo ou código notifique nesse canal.

Depois disso:

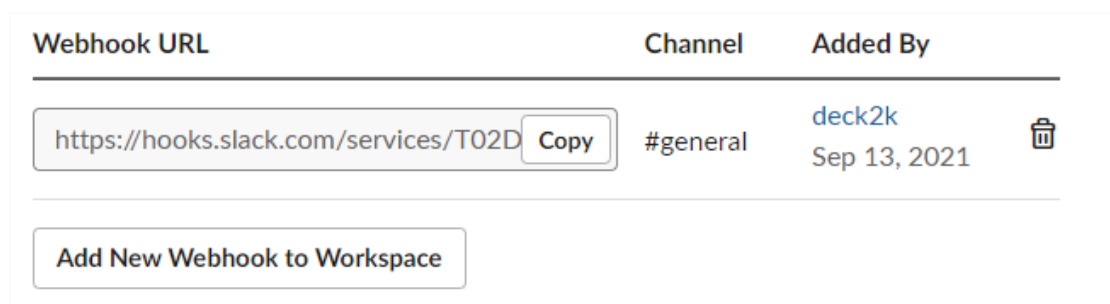
**Adicionar recursos e funcionalidade → Webhooks recebidos → ATIVE**



Uma vez ativado, vamos para a parte inferior da página e adicionamos um novo webhook, com a opção **Adicionar Novo Webhook ao Espaço de Trabalho**:



Ele nos perguntará em qual canal vamos autorizar este aplicativo a publicar as mensagens (podemos criar quantos webhooks quisermos). Depois de criado, veremos o seguinte:



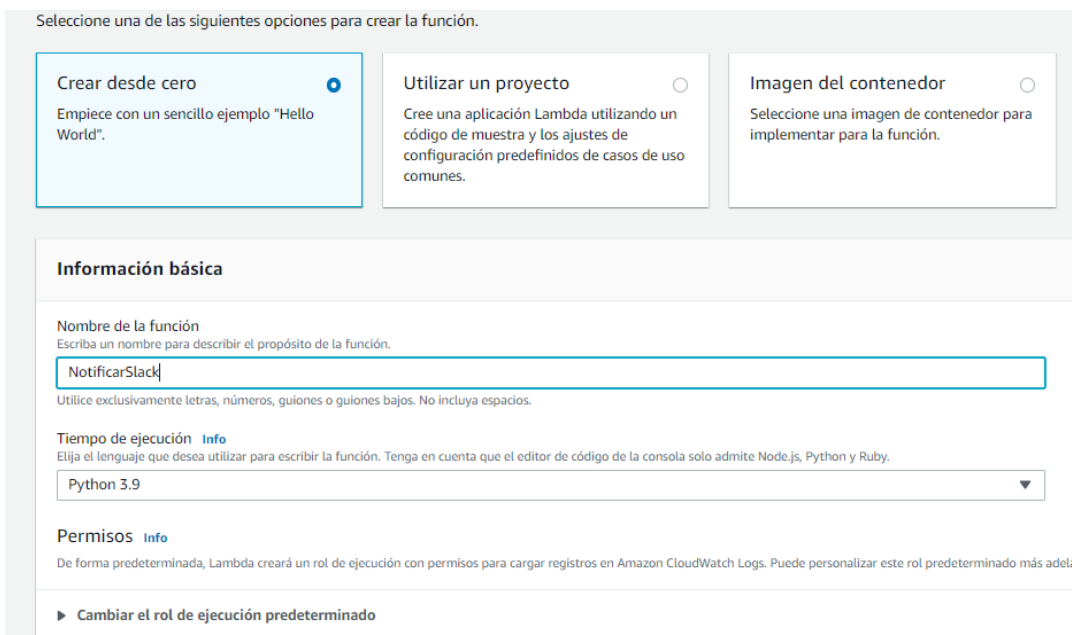
Percebemos que temos um **URL de Webhook**, nós o copiamos!

## 4. Integração com AWS Lambda

Já gerou o Webhook no Slack, precisamos invocá-lo do ecossistema AWS. Para isso vamos utilizar um dos seus serviços incluídos no denominado "Serverless": **AWS Lambda**. Este serviço nos permitirá adicionar lógica personalizada às nossas notificações CloudWatch.

Vamos para o serviço AWS Lambda e lá vamos "Criar uma nova função" e selecionar as opções:

- **Crie do zero, e como um tempo de execução, 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](#)

Uma vez lá, substituímos o código que aparece em **lambda\_function.py** pelo seguinte:

```
#!/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)
```

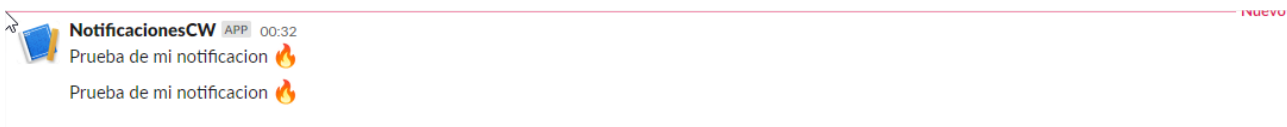
Acabamos de substituir o valor do url pelo URL do Slack Webhook da etapa 3:

```

1 #!/usr/bin/python3.6
2 import urllib3
3 import json
4 http = urllib3.PoolManager()
5 def lambda_handler(event, context):
6     url = "https://hooks.slack.com/services/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"
7     msg = {
8         "text": "Prueba de mi notificacion :fire:"
9     }
10    encoded_msg = json.dumps(msg).encode('utf-8')
11    resp = http.request("POST", url, body=encoded_msg)

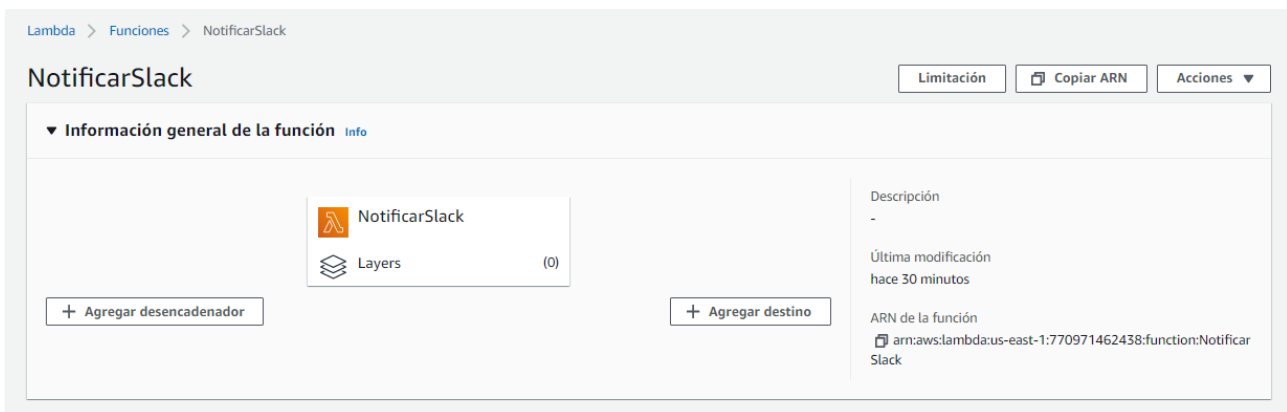
```

Uma vez feito isso, devemos implantar a função, e então podemos testá-la clicando em Testar. Se nossa integração for bem-sucedida, receberemos a mensagem no canal do Slack!



A última etapa é integrar esse recurso com o gatilho CloudWatch. Para fazer isso, vamos associá-lo ao tópico (ref. Página 10) criado nas notificações do CloudWatch.

No menu principal de nossa função Lambda, vamos "**Adicionar gatilho**":




Na seleção de serviços, procuramos o SNS e o selecionamos. Em seguida, escolhemos o tópico correspondente:

Lambda > Añadir desencadenador

## Añadir desencadenador

### Configuración del desencadenador

 **SNS**  
aws messaging notifications pub-sub push

Tema de SNS  
Seleccione el tema de SNS al que desea suscribirse.

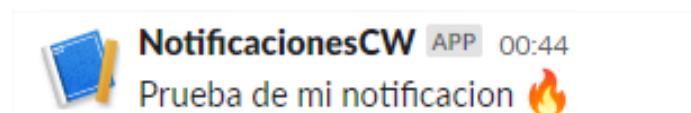
× ↺

Lambda añadirá los permisos necesarios para Amazon SNS para invocar la función Lambda desde este desencadenador.  
[Obtenga más información](#) sobre el modelo de permisos de Lambda.

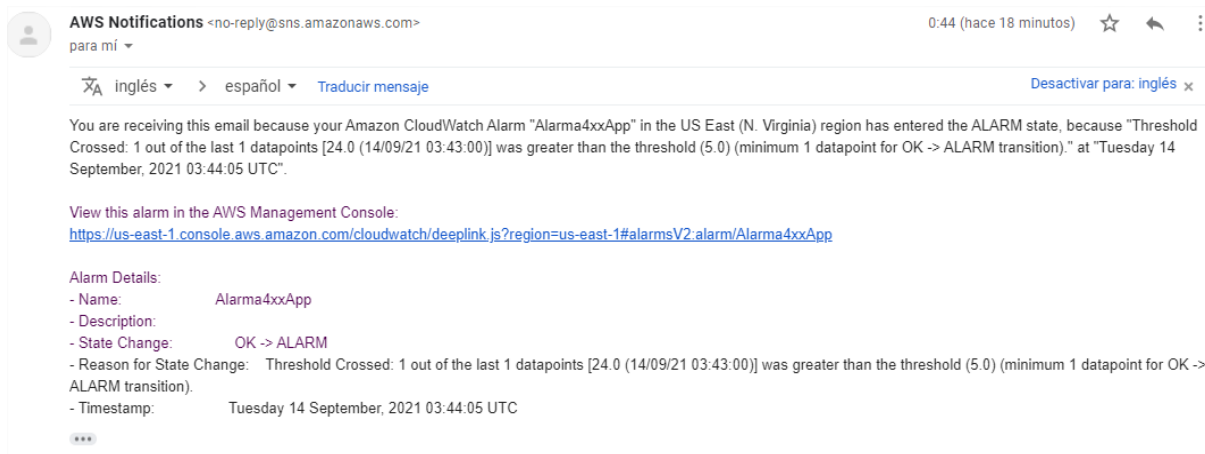
ⓘ The Lambda console no longer supports disabling SNS triggers. Delete these triggers to stop further actions.

Cancelar Agregar

Depois que essa etapa for concluída, podemos tentar novamente gerar solicitações que retornem um erro HTTP 4xx. Notemos que ele não apenas nos notificará por e-mail, mas também notificará o canal do Slack:



*Notificação do Slack*



### *Notificação de Email*

## Conclusão

Podemos concluir que as integrações possuem um espectro de aplicação que vai muito além do monitoramento.

Especificamente, a AWS facilita a troca de dados entre seus próprios serviços e produtos de terceiros, por meio da conectividade API. Desta forma, temos a opção de desenvolver, publicar, manter e monitorar em qualquer escala.

Com comunicação bidirecional e em tempo real, é um serviço poderoso e muitas vezes amplamente utilizado em ambientes modernos contendo microsserviços, sistemas distribuídos e um conceito que, até a data desta publicação, está em alta: a computação "sem servidor".

## Bônus

Convidamos você a ler e pesquisar sobre:

- "Arquitetura orientada a eventos" e "Arquitetura orientada a dados": [https://en.wikipedia.org/wiki/Event-driven\\_architecture](https://en.wikipedia.org/wiki/Event-driven_architecture)
- "Serverless Computing": [https://en.wikipedia.org/wiki/Serverless\\_computing](https://en.wikipedia.org/wiki/Serverless_computing)

Ambos os modelos, intimamente ligados à integração, são amplamente utilizados no mundo moderno.