



LINE BOT

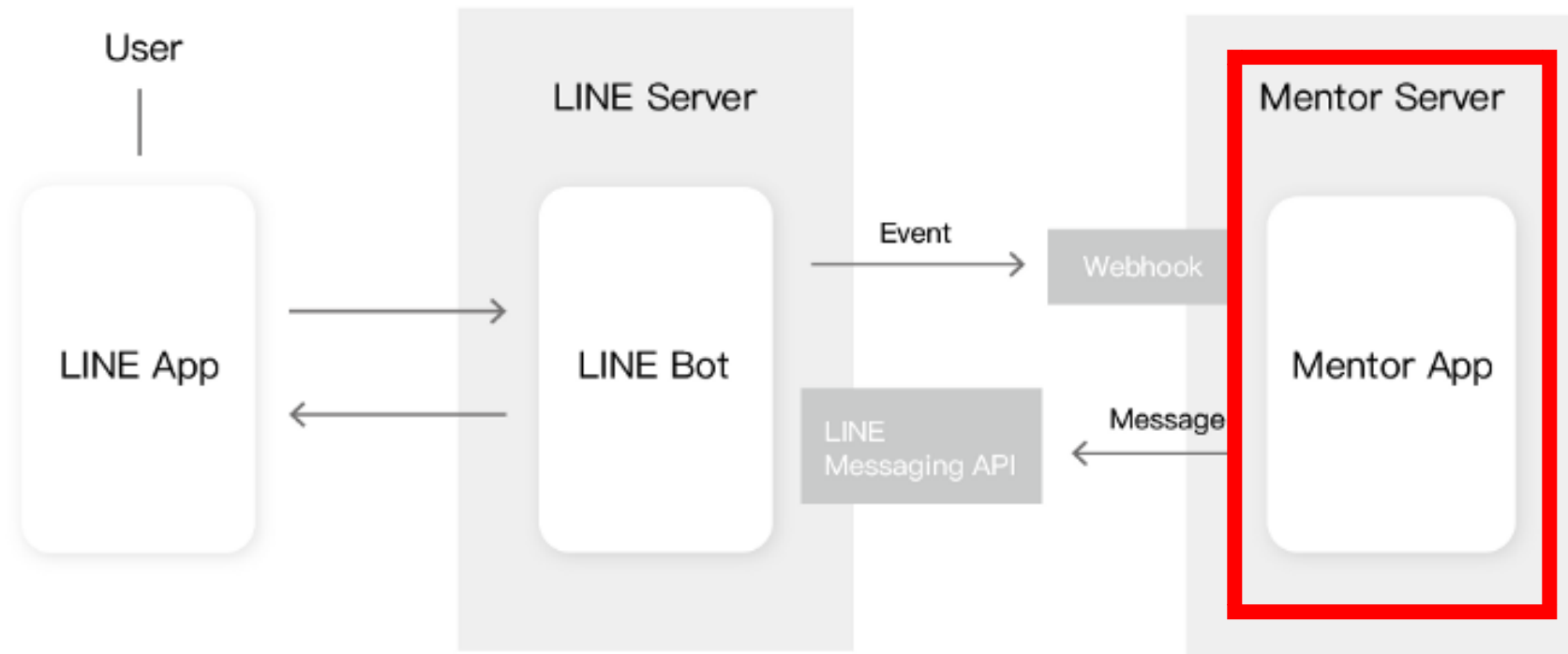
概念說明

- LINE Bot聊天機器人實際上是LINE的其中一個服務，叫做「LINE 官方帳號」。
- LINE Bot 機器人是在LINE 中一位特殊的機器人好友，其他 LINE 使用者（通常是客戶或對商品有興趣的潛在客戶）需要將此好友設定為朋友，才能夠與機器人進行互動。
- 我們能夠讓此機器人好友根據我們設計的程式，或透過後台管理，與其他LINE的使用者進行互動。

運作流程



AWS Lambda

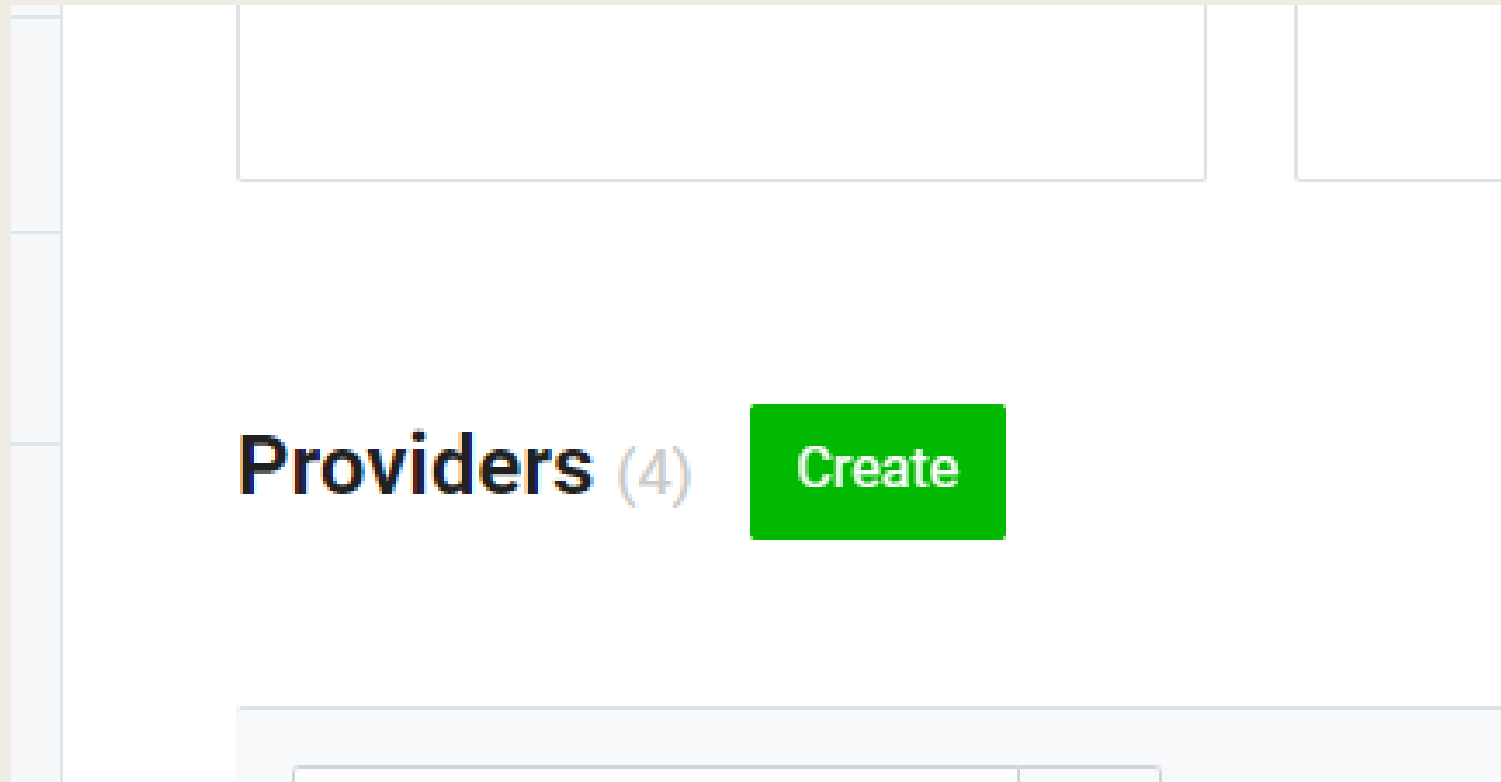


LINE 頻道建立(1)

- 要開發 LINE Bot 應用，首先要先申請 Line 開發者帳號。
- 開啟「<https://developers.line.biz/zh-hant/>」網頁，按 **Log in** 鈕進行登入。

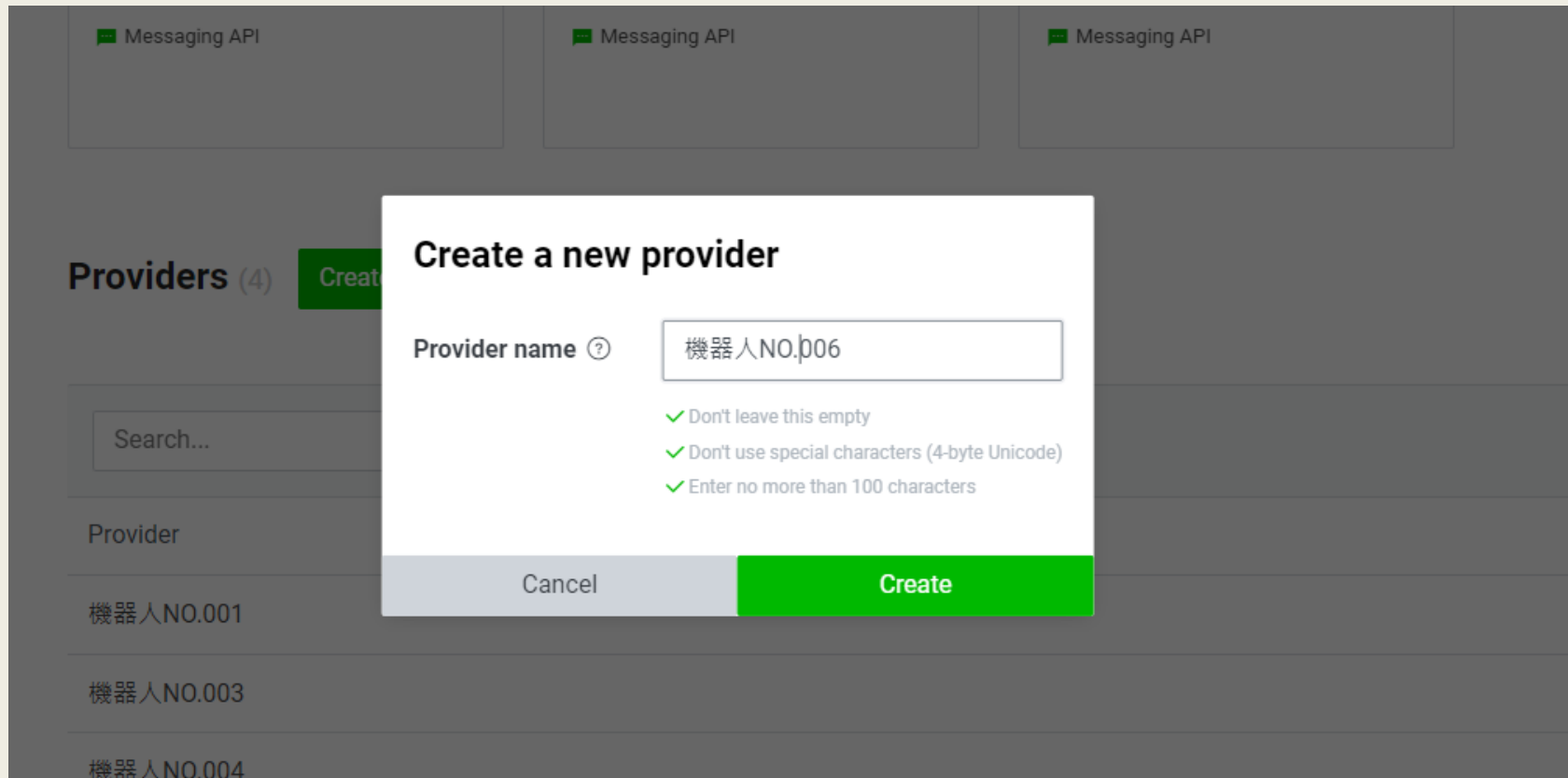
LINE 頻道建立(2)

- 接著建立 Provider，按 **Create** 鈕



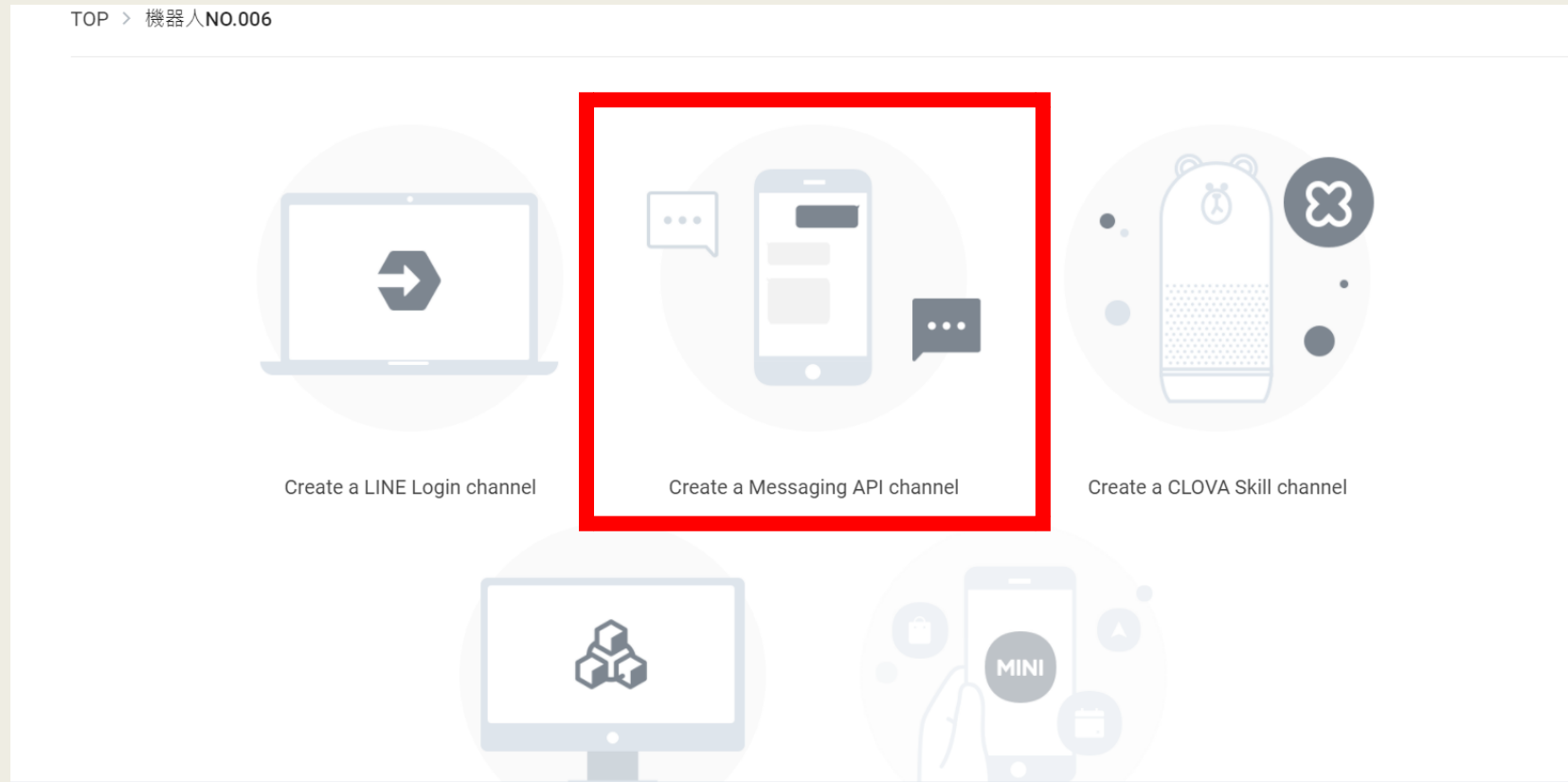
LINE 頻道建立(3)

- 輸入 Provider 名稱後，按 **Create** 鈕就新增一個 Provider



LINE 頻道建立(4)

- 建立 Provider 後，點選新建的Provider，再按 **Create a Messaging API channel** 建立一個LINE Bot。



LINE 頻道建立(5)

■ 輸入

- Channel name
- Channel description
- Category
- Subcategory

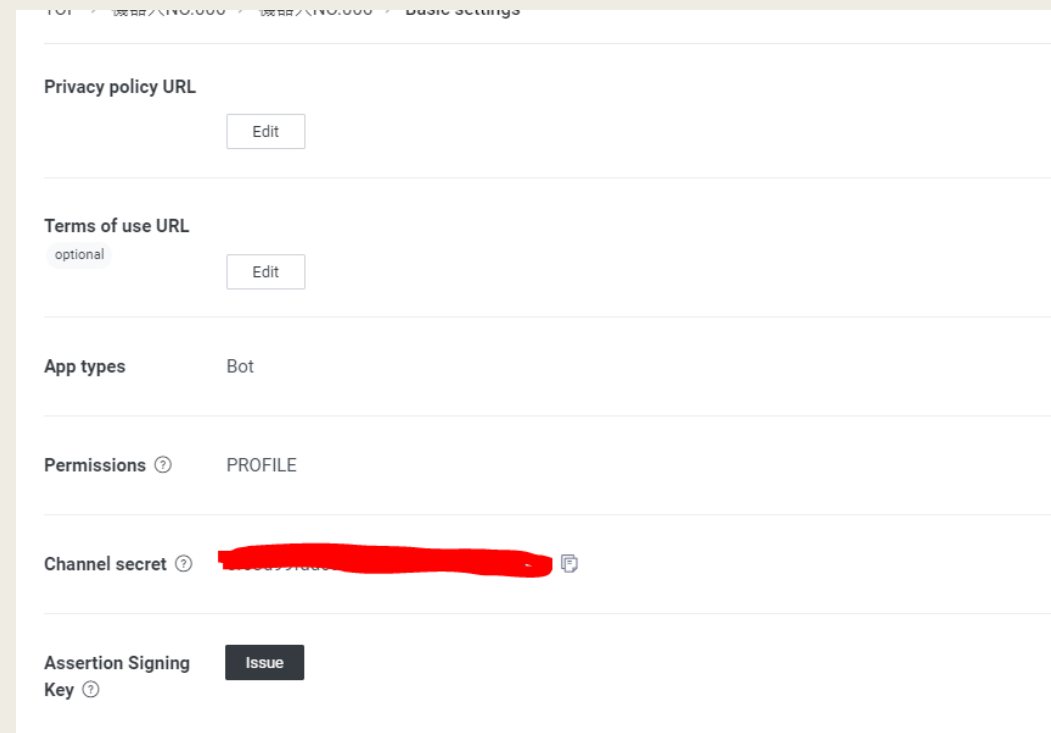
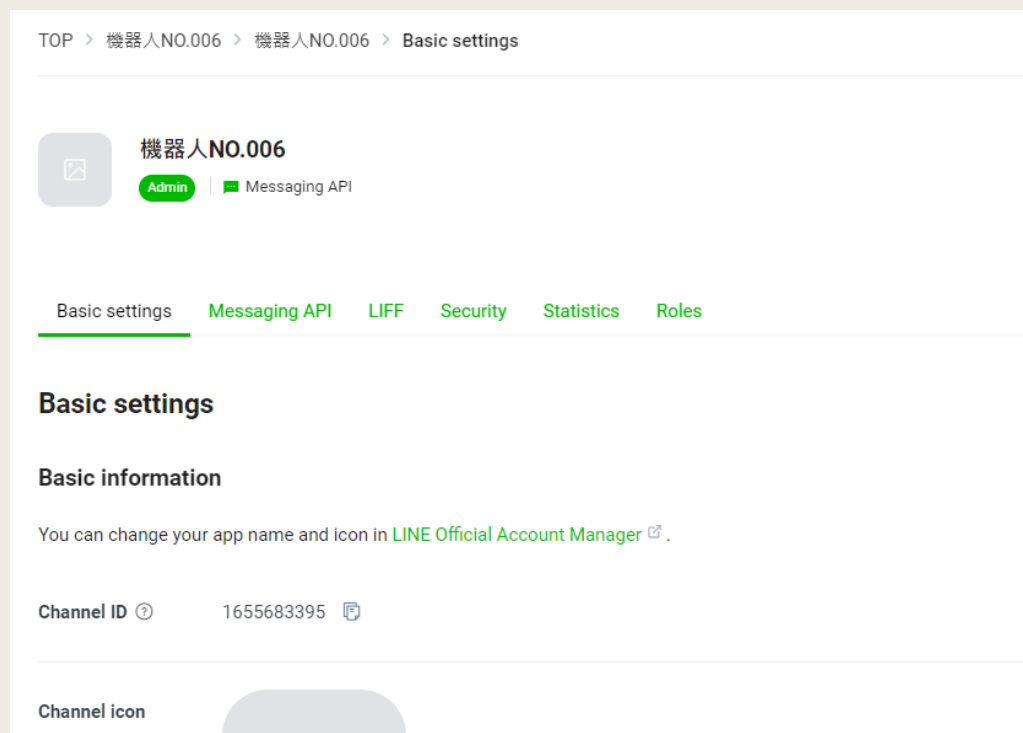
■ 勾選

- I have read and agree to the LINE Official Account Terms of Use
- I have read and agree to the LINE Official Account API Terms of Use

■ 以上完成後，按 **Create** 鈕

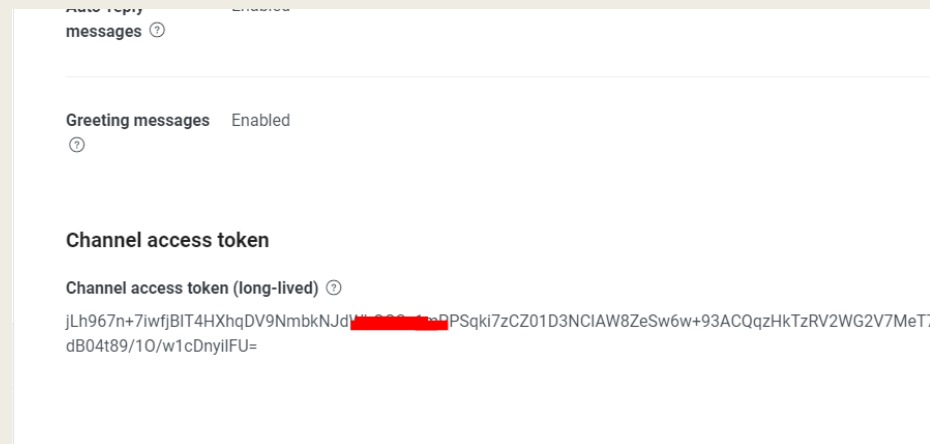
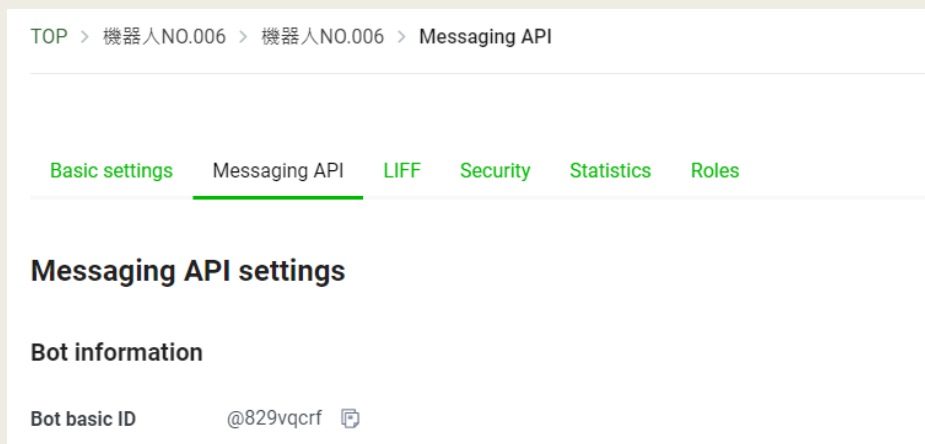
取得溝通憑證

- 創建好頻道後進入該頻道頁面，首先在 Basic settings 分頁中可以先記下 Channel secret 憑證密鑰會在之後用到。



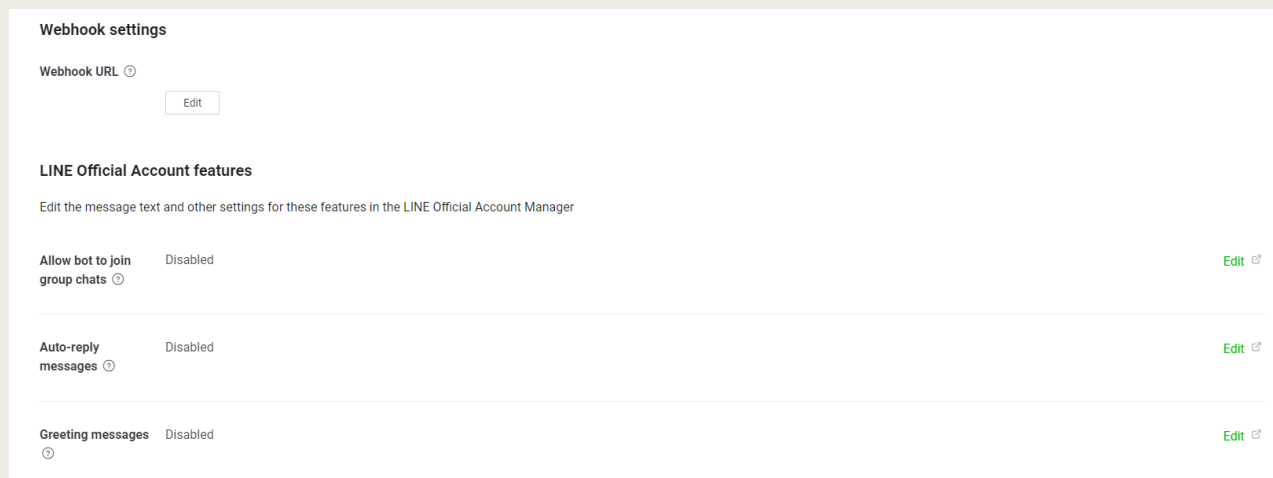
取得 Channel access token

- 接著 Messaging API 分頁下的 Channel access token (long-lived)
- 點擊「 Issue 」發佈一個新的憑證，系統會要求給予原先憑證的過期時間
- 新憑證產生後可以記下日後使用。



Webhook與自動回覆設定

- Webhook 是指負責接收聊天機器人發送事件的接口，也就是後續將談論的溝通程式，啟用方式是將 Use webhooks 改為 Enabled，
- 而 Webhook URL 則填寫接口位址，該值目前可以暫省略，待之後部署階段再來更新。
- 建議將 LINE Official Account features 分類下的設定 Auto-reply messages 和 Greeting messages 都改為 Disabled 關閉，避免與之後的信息回覆衝突。



The screenshot shows the 'Webhook settings' page in the LINE Official Account Manager. It includes a 'Webhook URL' field with an 'Edit' button. Below this, under 'LINE Official Account features', there are three settings: 'Allow bot to join group chats' (Disabled), 'Auto-reply messages' (Disabled), and 'Greeting messages' (Disabled). Each setting has an 'Edit' button with an external link icon.

Setting	Status	Action
Webhook URL	(Empty)	Edit
Allow bot to join group chats	Disabled	Edit
Auto-reply messages	Disabled	Edit
Greeting messages	Disabled	Edit



The screenshot shows the 'Response settings' page in the LINE Official Account Manager. It is divided into 'Basic settings' and 'Advanced settings'. In 'Basic settings', 'Response mode' is set to 'Chatbot' (selected), and 'Add welcome message for friends' is set to 'Stop' (selected). In 'Advanced settings', 'Automatic response message' is set to 'Stop' (selected), and 'Webhook' is set to 'Enabled' (selected).

Setting	Status	Action
Response mode	Chatbot (selected)	
Add welcome message for friends	Stop (selected)	加入好友的歡迎訊息設定
Automatic response message	Stop (selected)	自動回應訊息設定
Webhook	Enabled (selected)	Messaging API設定

LINE SDK

- 對於初學者而言，最快能建立一個LINE聊天機器人的方法應該就是套用LINE Bot SDK。LINE Bot SDK提供各種常見程式語言的函式庫，讓開發者很輕易便能開發LINE Messaging API的應用程式。這些SDK還包括範例程式，讓初學者能更快瞭解如何使用函式庫。以下是LINE Bot SDK所支援的程式語言列表。

- [Java](#)
- [PHP](#)
- [Go](#)
- [Perl](#)
- [Ruby](#)
- [Python](#)
- [Node.js](#)

建立 Lambda

[Lambda](#) > [函式](#) > 建立函式

建立函式 [Info](#)

選擇下列選項之一來建立您的函式。

從頭開始撰寫 ☒

從簡單的 Hello World 範例開始。

使用藍圖 ☐

透過常用案例的範本程式碼與組態預設，建置 Lambda 應用程式。

容器映像 ☐

攝取要有函數部署的容器映像。

瀏覽無伺服器應用程式儲存庫 ☐

透過 AWS Serverless Application Repository 部署範例 Lambda 應用程式。

基本資訊

函式名稱
輸入描述函式目的之名稱。

linebot-python

只能使用字母、數字、連字號或底線，不得有空格。

執行時間 [Info](#)
選擇要用來撰寫函式的語言。請注意，主控台程式碼編輯器僅支援 Node.js、Python 和 Ruby。

Python 3.8

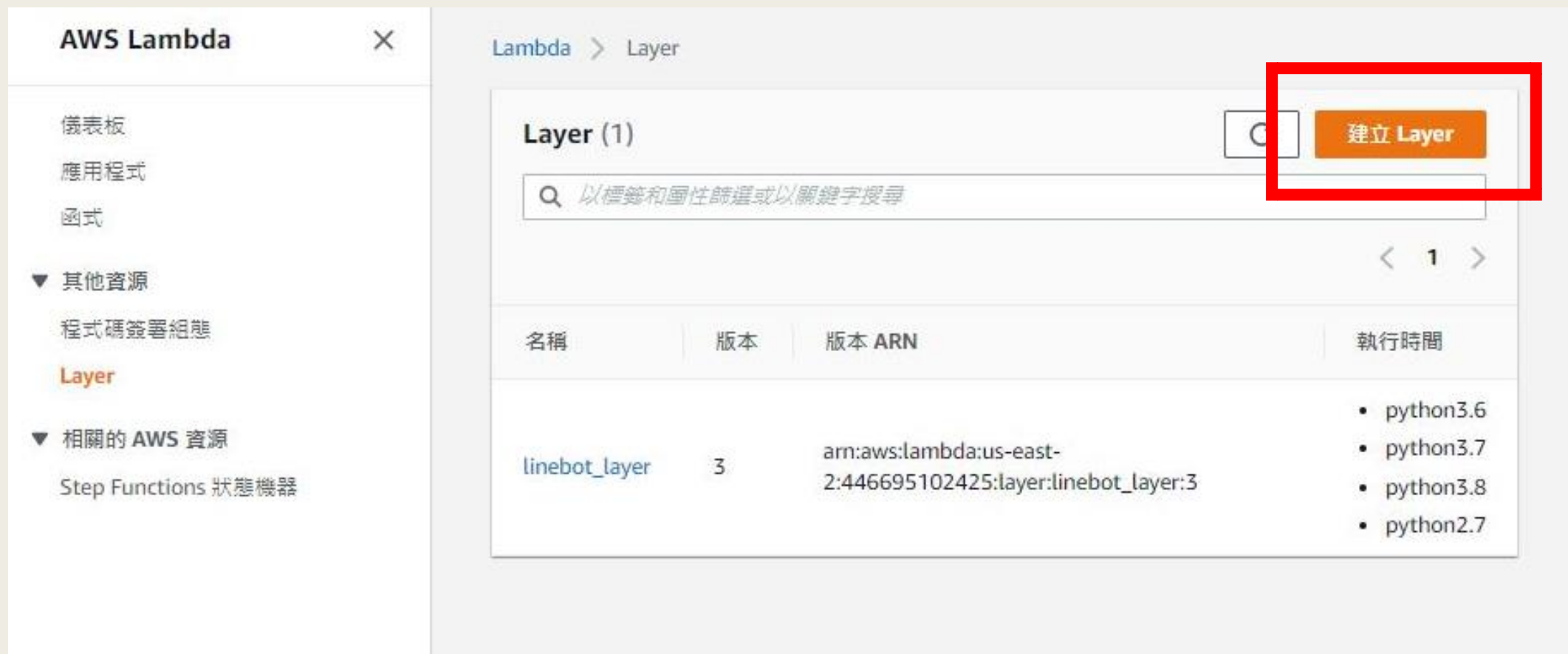
許可 [Info](#)
預設情況下，Lambda 會建立具有將日誌上傳至 Amazon CloudWatch Logs 許可的執行角色。您可以在稍後新增開發條件時，自訂此預設角色。

► 變更預設執行角色

► 進階設定

取消 建立函式

建立 Lambda Layer



The screenshot shows the AWS Lambda console interface. On the left is a navigation sidebar with the following items: 儀表板, 應用程式, 函式, 其他資源 (expanded), 程式碼簽署組態, Layer (highlighted in orange), and 相關的 AWS 資源 (expanded) with Step Functions 狀態機器. The main content area is titled 'Lambda > Layer'. At the top right of this area, there is a search bar and a red box highlighting the '建立 Layer' button. Below the search bar is a table with the following columns: 名稱, 版本, 版本 ARN, and 執行時間. The table contains one entry: 'linebot_layer' with version '3' and ARN 'arn:aws:lambda:us-east-2:446695102425:layer:linebot_layer:3'. The '執行時間' column for this entry lists: python3.6, python3.7, python3.8, and python2.7.

Layer (1)

以標籤和屬性篩選或以關鍵字搜尋

名稱	版本	版本 ARN	執行時間
linebot_layer	3	arn:aws:lambda:us-east-2:446695102425:layer:linebot_layer:3	<ul style="list-style-type: none">python3.6python3.7python3.8python2.7

建立 Lambda Layer

Lambda > Layer > 建立 Layer

建立 Layer

建立新 Layer

名稱

描述 - 選用

☒ 上傳 .zip 檔案
☐ 上傳來自 Amazon S3 的檔案

python.zip (2.2 MB)

大於 10 MB 的檔案，請考慮使用 Amazon S3 上傳。

相容的執行時間 - 選用 [Info](#)
選擇至多 15 個執行時間。

Python 2.7 ✕

Python 3.6 ✕

Python 3.7 ✕

Python 3.8 ✕

授權 - 選用 [Info](#)

取消

建立

使用 Lambda Layer

Lambda > 函式 > linebot-python

linebot-python

調節 複製 ARN 操作 ▼

▼ 函式概觀 Info

+ 新增觸發


linebot-python

 Layers (0)

+ 新增目的地

描述
-

上次修改時間
21 秒前

函數 ARN
 arn:aws:lambda:us-east-2:446695102425:function:linebot-python

程式碼 測試 監控 組態 別名 版本

使用 Lambda Layer

執行時間
Python 3.8

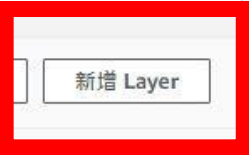
處理常式 [Info](#)
lambda_function.lambda_handler

Layer [Info](#)

編輯 新增 Layer

合併順序	名稱	Layer 版本	版本 ARN
沒有資料可以顯示。			

程式碼簽署詳細資訊 [Info](#)



使用 Lambda Layer

Lambda > Layer > 新增 Layer

新增 Layer

選擇 Layer [Info](#)

選擇執行時間相容的 Layer，或指定 Layer 版本的 Amazon Resource Name (ARN)。



AWS Layer

從 AWS 提供的 Layer 清單中選擇 Layer。



自訂 Layer

從您的 AWS 帳戶或組織建立的 Layer 清單中選擇 Layer。



指定 ARN

提供 ARN 以指定 Layer。

自訂 Layer

由您的 AWS 帳戶或組織建立，且與您函式執行時間相容的 Layer。

linebot-python



版本

1



取消


新增


建立 API Gateway

linebot-python

調節

▼ 函式概觀 Info

 linebot-python

 Layers (1)

+ 新增觸發

+ 新增目的地


描述

-

上次修改時間

24 秒前

函數 ARN

 arn:aws:lambda:us-east-2:446695102:python

程式碼

測試

監控

組態

別名

版本

建立 API Gateway

Lambda > 新增 觸發條件

新增 觸發條件

觸發組態

 API Gateway
api application-services aws serverless

將 API 新增至您的 Lambda 函數，以建立可叫用函數的 HTTP 端點。API 閘道支援兩種類型的 RESTful API：HTTP API 和 REST API。 [進一步了解](#)

API
建立新的 API 或連接現有的 API。

建立 API

API 類型

☐ HTTP API
建立 HTTP API。

☒ REST API
建立 REST API。

安全性

設定您的 API 端點的安全性機制。

開啟

請勿新增任何授權或身份驗證要求。任何使用者都可以透過 HTTP 叫用來叫用您的函數。

▶ 其他設置

Lambda 會增加 Amazon API Gateway 的必要權限，以使用這個觸發條件叫用您的 Lambda 函式。 [進一步了解 Lambda 權限模型](#)。

取消 新增

建立 API Gateway


The screenshot displays the AWS API Gateway console interface. At the top, there's a header with the title '觸發 (1)' and several action buttons: '刷新', '啟用', '停用', '修正錯誤', '刪除', and '新增觸發'. Below the header is a search bar labeled '尋找觸發' and a pagination indicator showing '< 1 >'. The main content area is titled '觸發條件' and contains a list of triggers. The first trigger is 'API Gateway : linebot-python-API', which is expanded to show its details. The details include: '方法 : ANY', '安全性 : NONE', '授權 : NONE', '啟用指標和錯誤記錄 : 否', '階段 : default', '資源路徑 : /linebot-python', 'API : api-gateway/gfprep9g15/*/*/linebot-python', and 'API 終端節點 : https://gfprep9g15.execute-api.us-east-2.amazonaws.com/default/linebot-python'. The last line is highlighted with a red rectangular box.

觸發 (1)

刷新 啟用 停用 修正錯誤 刪除 新增觸發

Q 尋找觸發 < 1 >

☐ 觸發條件

 **API Gateway : linebot-python-API**
arn:aws:execute-api:us-east-2:446695102425:gfprep9g15/*/*/linebot-python

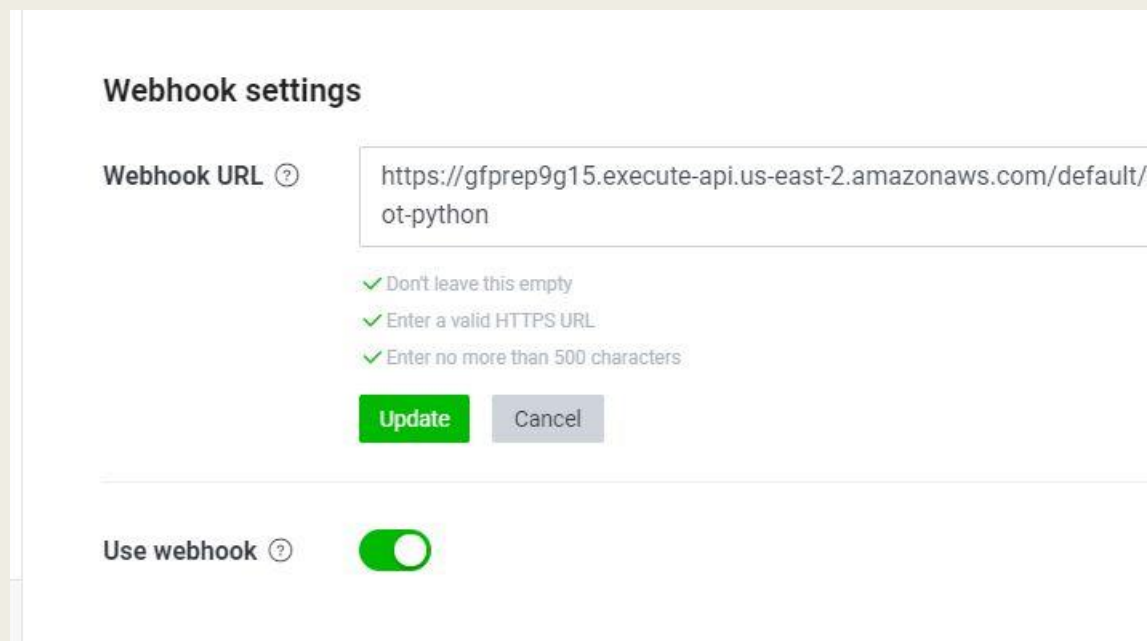
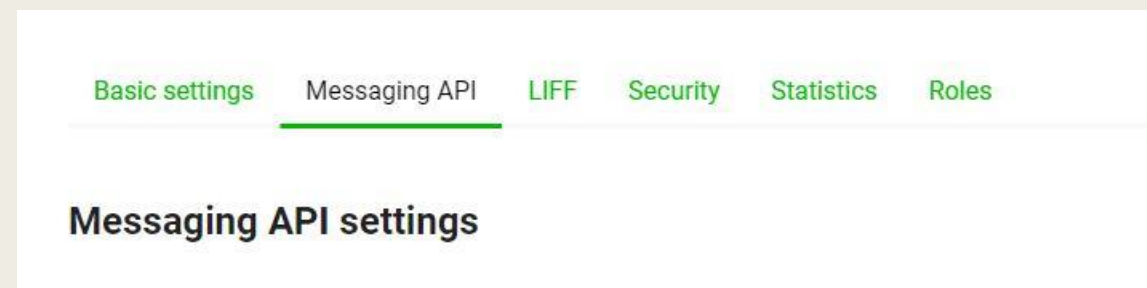
▼ 詳細資訊

方法 : ANY
安全性 : NONE
授權 : NONE

☐ 啟用指標和錯誤記錄 : 否
階段 : default
資源路徑 : /linebot-python
API : api-gateway/gfprep9g15/*/*/linebot-python
API 終端節點 : <https://gfprep9g15.execute-api.us-east-2.amazonaws.com/default/linebot-python>

設定 Webhook

- 在 Messaging API 分頁下
- 將剛取得的”API 終端節點”，輸入 Webhook URL



Lambda 環境變數

The screenshot displays the AWS Lambda console interface for a function named 'linebot-python'. The 'API Gateway' tab is selected, and the '新增觸發' (Add Trigger) button is visible. The '環境變數' (Environment Variables) tab is highlighted with a red box. The left sidebar shows the '環境變數' option selected with a red box. The main content area shows '環境變數 (0)' (0 Environment Variables) and a table with columns '金鑰' (Key) and '值' (Value). A message states '沒有環境變數' (No environment variables) and '沒有與此函數相關聯的環境變數。' (No environment variables are associated with this function). A red box highlights the '編輯' (Edit) button.

API Gateway

+ 新增觸發

+ 新增目的地

1 小時前

函數 ARN

arn:aws:lambda:us-east-2:446695102425:function:linebot-python

程式碼 | 測試 | 監控 | **環境變數** | 別名 | 版本

一般組態

觸發

許可

目的地

環境變數

環境變數 (0)

編輯

金鑰	值
沒有環境變數	
沒有與此函數相關聯的環境變數。	

編輯

Lambda 環境變數

Lambda > 函式 > linebot-python > 編輯環境變數

編輯環境變數

環境變數

您可以將環境變數定義成可從您的函式程式碼存取的鍵值組，這些鍵值組可用來備存組態設定，而不需要變更函式程式碼。
[進一步了解](#)

此函式沒有環境變數。

新增環境變數

► 加密設定

取消 儲存

Lambda 環境變數

Lambda > 函式 > linebot-python > 編輯環境變數

編輯環境變數

環境變數

您可以將環境變數定義成可從您的函式程式碼存取的鍵值組。這些鍵值組可用來儲存組態設定，而不需要變更函式程式碼。
[進一步了解](#)

金鑰

YOUR_CHANNEL_ACCESS_TOKEN

值

QxguwQYZoECaUx6ftvtZou97cB23hcBc

移除

YOUR_CHANNEL_SECRET

a3efbb2af6c4ea6cad086a78a4b7737f

移除

新增環境變數

► 加密設定

取消

儲存

簡單回應

程式碼

測試

監控

組態

別名

版本

程式碼來源 Info

上傳於 ▼

File Edit Find View Go Tools Window

Test

Deploy

Changes deployed

Go to Anything (Ctrl-P)

Environment

linebot-python - /

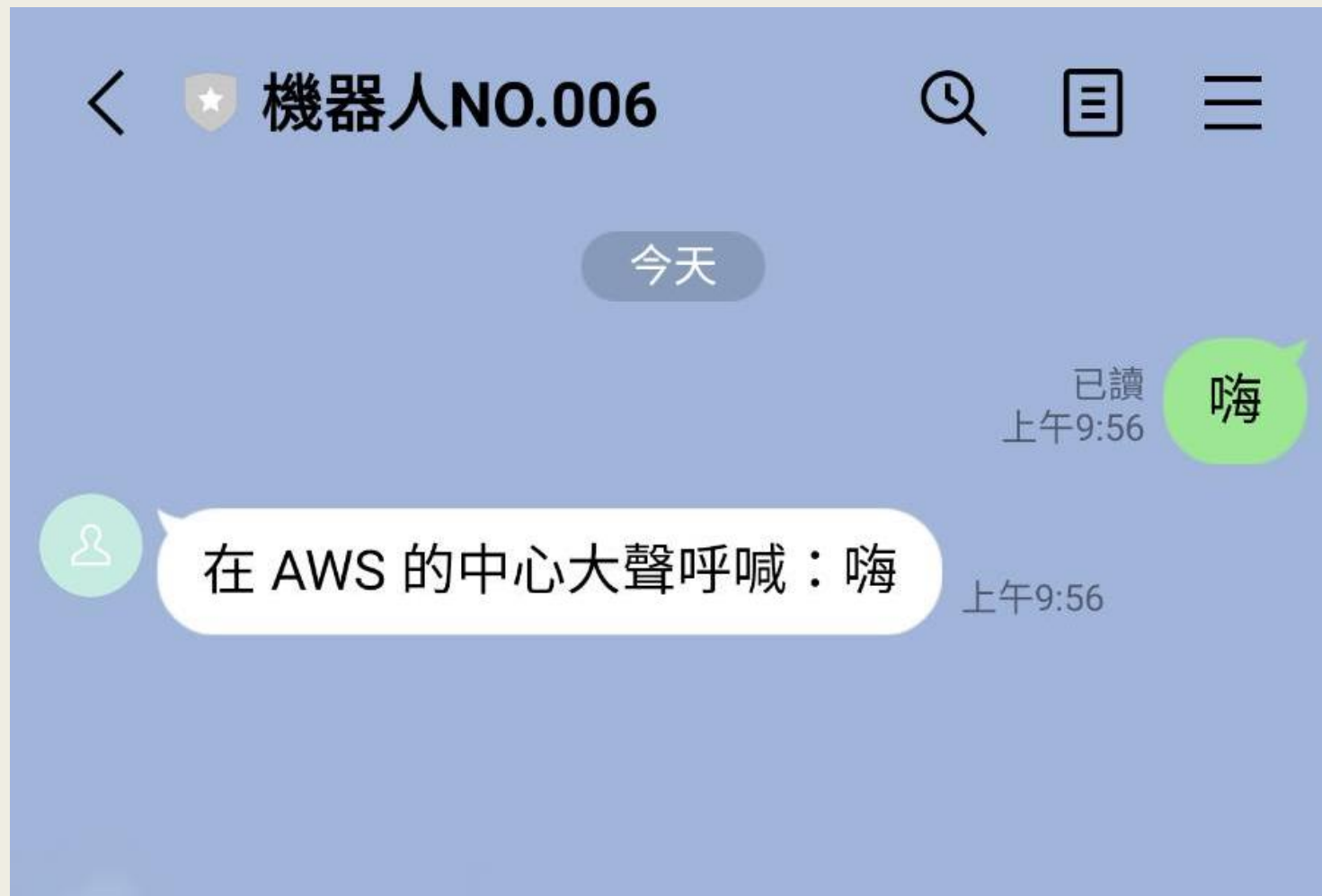
lambda_function.py

lambda_function

```
1 import os
2
3 from linebot import LineBotApi, WebhookHandler
4 from linebot.exceptions import InvalidSignatureError
5 from linebot.models import *
6
7 # 初始化 LineBotApi 和 WebhookHandler
8 line_bot_api = LineBotApi(os.environ['YOUR_CHANNEL_ACCESS_TOKEN'])
9 handler = WebhookHandler(os.environ['YOUR_CHANNEL_SECRET'])
10
11 def lambda_handler(event, context):
12     @handler.add(MessageEvent, message=TextMessage)
13     def handle_message(event):
14         if event.source.user_id != "Udeadbeefdeadbeefdeadbeefdeadbeef":
15             line_bot_api.reply_message(event.reply_token, TextSendMessage(text='在 AWS 的中心大聲呼喊：' + event.message.text))
16
17     try:
18         # get X-Line-Signature header value
19         copy_headers = {key.lower(): value for key, value in event['headers'].items()}
20         signature = copy_headers['x-line-signature']
21
22         # get request body as text
23         body = event['body']
24
25         # handle webhook body
26         handler.handle(body, signature)
27
28     except InvalidSignatureError:
29         print("Invalid signature. Please check your channel access token/channel secret.")
30         return {'statusCode': 400, 'body': 'InvalidSignature'}
31
32     except Exception as e:
33         print(e)
34         return {'statusCode': 400, 'body': str(e)}
35
36     return {'statusCode': 200, 'body': 'OK'}
```

3:47 Python Spaces: 4

簡單回應



參考資料

- [LINE Bot by Python 全攻略：從Heroku到AWS跨平台實踐（iT邦幫忙鐵人賽系列書）](#)
- [Python與LINE Bot機器人全面實戰特訓班](#)
- [輕鬆學會LINE程式設計與AI聊天機器人實作開發](#)
- <https://tw.linebiz.com/download/line-official-account>
- [LINE Bot 聊天機器人 #1：從認識開始](#)
- [LINE Bot 聊天機器人 #2：創建頻道](#)
- [LINE 2.0 官方帳號收費新制上線！2021企業社群經營該如何因應？](#)
- [開發LINE聊天機器人不可不知的十件事](#)
- [LINE Bot 系列文 — 什麼是 Webhook?](#)