使用 AWS Lambda
(搭配 AWS API Gateway)
快速部署 LINE Bot

文章撰寫 alvin.constantine@outlook.com

什麼是 AWS Lambda? 用它來部署 Line Bot 有什麼不同和好處?

讓你專注在應用程式開發，不用架設伺服器、設定監控、網路安全性及建設基礎架構

它就像是 *Python* 中的 一個 *Lambda Function* 。這是 *Lambda* 這個服務的簡單概念。

- 應用程式即是一個 Function，最長可執行 15 分鐘。
- 應用程式執行的前與後，服務為停止狀態。

把應用程式寫成可執行的 *Function*，其他由雲服務代管。這是 *Function As a Service* 的簡單概念。

- 雲服務提供 *Function* 的執行環境 (*Runtime*)，執行你包裝好的 *Function* 應用程式。
- 應用程式的 Function 只存在執行階段 (*Runtime*)，執行環境終止也代表所有資料不會留存。

不用執行或架設任何伺服器，應用程式只會使用執行階段的運算時間。這是 *Serverless* 的簡單概念。

- 不用擔心或管理任何伺服器。由雲服務負責所有的其他工作；除了你開發的應用程式。
- 使用雲服務提供的 *Free-Tier* 方案。在方案提供的限額內永久免費。

用 Lambda 來部署 Line Bot 我需要具備什麼條件?

註冊的 *LINE* 與 *AWS* 服務帳號，以及與你要開發的應用程式相關的知識及其商業邏輯

註冊的 *AWS* 雲服務帳號

- 可以註冊 *AWS Free-Tier* 方案: *https://aws.amazon.com/tw/free/*，提供一年有限度的免費服務。
- 註冊帳號需要有可收到信的 *email* 帳號。需要提供有效的信用卡卡號 (驗證 USD$1 但不會收費)。
- 註冊帳號過程為全自動化，過程中請確認輸入正確的資料，註冊完後服務即可使用。

註冊的 *LINE* 服務帳號

- 在 LINE 的開發網站建立一個 LINE Provider 和 Message API Channel。
- 啟用 LINE Message API 的 webhook。稍後會說明細節。

對 *Python* 程式語言或其他 AWS Lambda 支援的程式語言有基本的認識與瞭解

- 也可以參考雲服務商及 LINE 服務商提供的 SDK 範例，協助你開發 LINE Bot 應用程式。

有什麼注意事項?

注意安全性的控管，並關注 *AWS Free-Tier* 的免費限額！

安全性

- 雲服務廠商專注於基礎架構及服務本身的安全性，使用者需自行控管應用程式本身的安全性。
- 你使用的 email 帳號與密碼及註冊帳號的密碼等安全性保護也很重要。
- 如果疏於管理安全性，可能會導致服務被惡意使用造成鉅額的使用費用！

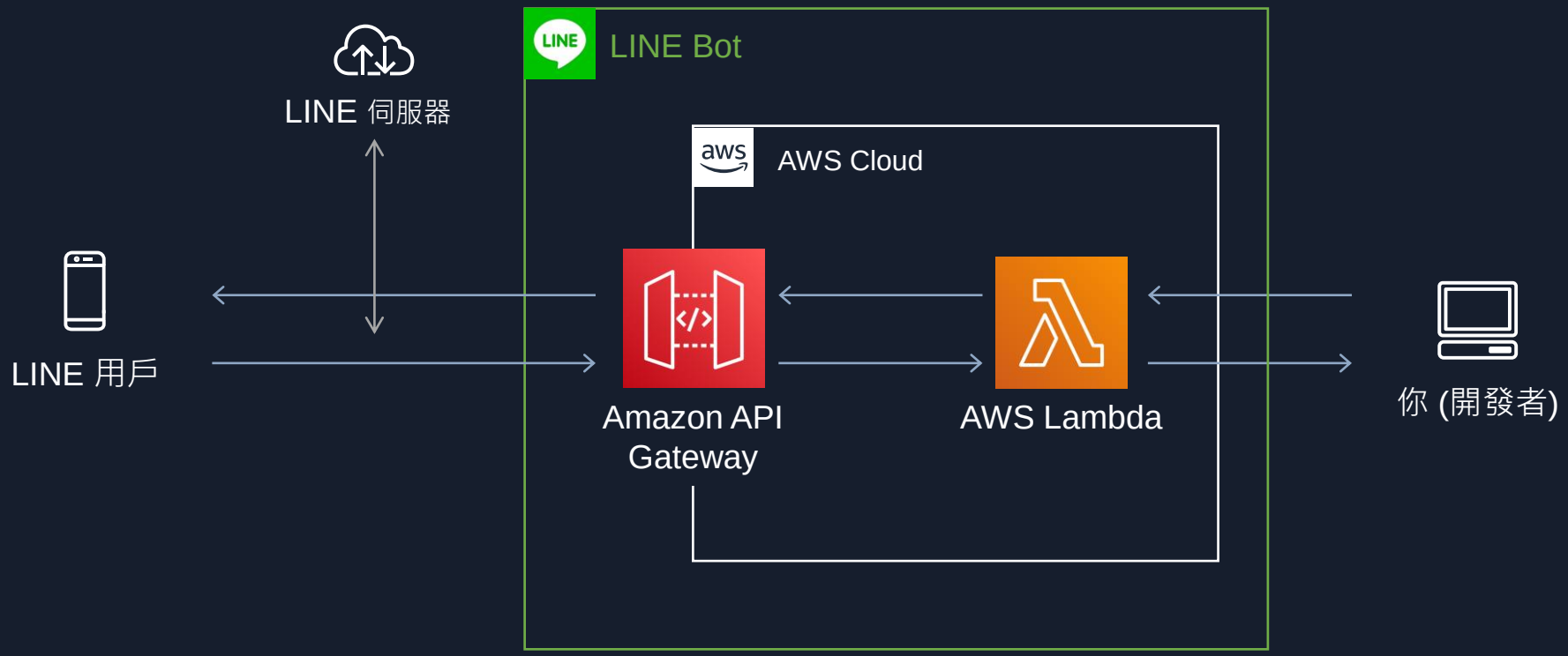
查看 *AWS Free-Tier* 的細節

- AWS 的 *Free-Tier* 免費方案，提供 AWS Lambda 每月 1 百萬個免費請求(*request*)。
- 如果需搭配使用其他雲服務儲存 AWS Lambda 執行運算完的資料，需注意相關的費用與成本。

設定 AWS 費用的郵件提醒通知並定期查看 AWS 管理頁面的費用細節

- 即時收到費用通知，瞭解使用狀況。並在需要時與雲服務支援人員保持連繫以瞭解細節。

架構簡圖概觀



步驟 1 | 步驟 2 | 步驟 3

建立 Lambda Function | 上傳打包好的應用程式 | 修改 LINE API Webhook

步驟 1. 登入 AWS 管理界面建立 Lambda Function

預設的帳號為 Root Account 有最高的權限。可參考 AWS 文件啟用雙階段認證!

使用瀏覽器開啟 *https://console.aws.amazon.com/*

- 使用註冊的 email 帳號和密碼登入，或使用另外建立的 Administrative Access 的 IAM 帳號登入
- 登入後可以在右上角先選擇區域。可以使用 東京 (ap-northeast-1) 或 香港 (ap-east-1) 離台灣較近。
- 需注意香港 (ap-east-1) 區域需在登入後額外啟用該區域。請參考 AWS 網站的說明。

點取左上角的 Services，點選 運算類別的 Lambda 服務開啟 Lambda 服務界面。按一下建立函式

- 選最左邊的從頭開始撰寫，為函式命名。選 Python 3.7 為執行時間，其他預設值不變，選**建立函式**。

在新增觸發條件選按一下，選 API Gateway，Create API，HTTP API，安全性為開啟。按一下新增。

- 點選 API Gateway 下方詳細資訊，將 API 終端節點旁的網址填入 LINE Bot 的 Webhook (參考步驟 3)

步驟 2. 打包應用程式及相依套件上傳到 AWS Lambda

為簡化細節，此文件提供基礎的範例套件。僅需上傳一次。爾後直接在界面編寫程式

點畫面中間的 Lambda 圖示，在右側的操作上點一下選取上傳 .zip 檔案。

- 預設的函式左邊只有一個根目錄和一個 *.py* 檔案。點選 *.py* 檔案後右側即是該檔案的程式碼內容。
- 按上傳後選取隨附的 LINE Bot 及相關套件範例 *LineBot_AWS_Lambda_QuickEx.zip*，然後按儲存。
- 上傳後的程式及套件會覆蓋原有的預設執行環境。如果已先在剛才的環境撰寫程式需先匯出備份。
- 如果上傳超過 3MB 的 .zip 會有訊息提醒你 Lambda 編輯畫面無法使用。需使用 AWS CLI 等工具。

將 LINE Bot 的 Channel Access Token 及 Channel Secret 新增到程式碼下方的環境變數 (參考步驟 3)

- 由於安全性的考量，這裡採用環境變數的方式儲存這兩個設定，再從程式碼裡用 *os.getenv* 去存取。
- 環境變數預設是自動使用 AWS KMS 加密的。在免費方案中採用 KMS 有提供有限的免費用量。

可以直接線上在中間畫面中修改你的 LINE Bot 程式。修改後按一下 Deploy 即可部署最後修訂的內容。

- 在 AWS Lambda 裡也有提供發行新版本或匯出函式等多樣性的功能。

步驟 3. 修改 LINE Message API Webhook 指向 AWS API Gateway

你的 Line Bot 應用程式已部署完成。讓 LINE Server 指向你的 Line Bot 產生關聯

開啟 LINE 的開發頁面 *https://developers.line.biz/console/* 並登入你的 LINE 帳號建立 Provider

- 建立 Provider 為建立 Line Bot 的第一個步驟。建立完後選擇該 Provider 後點選建立 Channel。

在建立 Channel 裡選擇建立 Message API，並啟用 Use webhook

- 建立 Message API 的後，點選看到 Basic Settings 設定。在頁面中找到 Channel secret 的值。
- 在下一個 Messaging API 頁籤中找到 Channel access token，點 Issue，取得 token 的值。
- 回到 Basic Settings 按一下 LINE Official Account Manager 的連結，選取左邊 Message API。
- 在 Webhook 網址中填入步驟 1 中的 API Gateway 下方詳細資料 的 API 終端節點。按儲存。
- 回 LINE Developers 頁面 Messaging API 頁籤，找到 Webhook settings，將 Use webhook 啟用。

完成啦~ 這樣你的 LINE Bot 就可以使用囉！ 是不是很容易呢?

- 在 Messaging API 頁籤中找 Bot information ，打開手機的 LINE 掃描 QR Code 加入好友即可對話。

其他參考資訊

如何打包套件？

- 建立一個資料夾。使用 pip install *Package* -t . 。

開發 LINE Bot 時碰到錯誤或問題如何除錯 (Debug/Troubleshooting)？

- 所有的記錄包含 Print() 函式的內容會包含在 CloudWatch 的 Log Group 裡。請參考後面的螢幕擷圖。

建立您的第一個 IAM 管理員用戶和群組

- *https://docs.aws.amazon.com/zh_tw/IAM/latest/UserGuide/getting-started_create-admin-group.html*

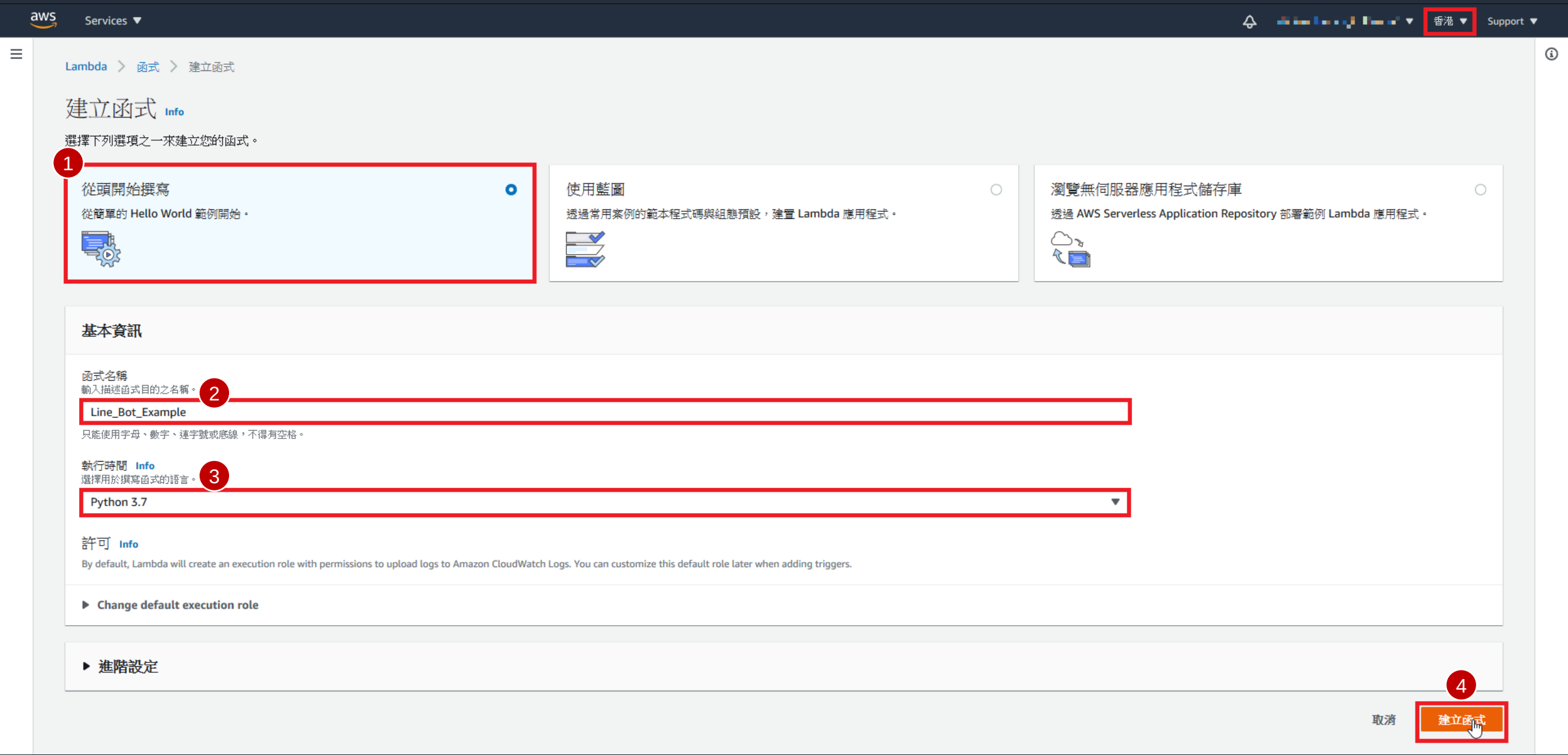
AWS Lambda 常見問答集

- *https://aws.amazon.com/tw/lambda/faqs/*

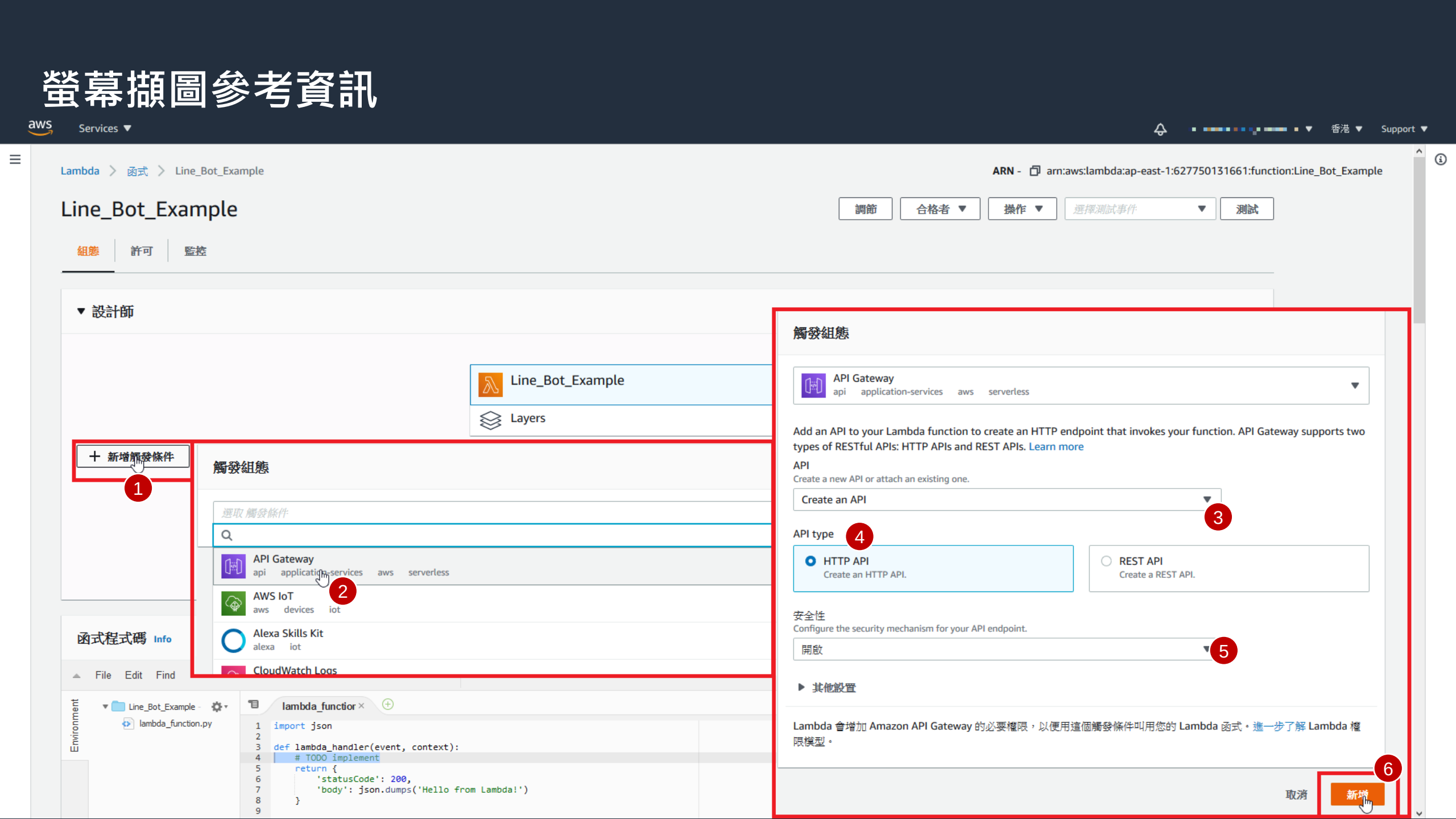
LINE Bot SDK for Python

- *https://github.com/line/line-bot-sdk-python*

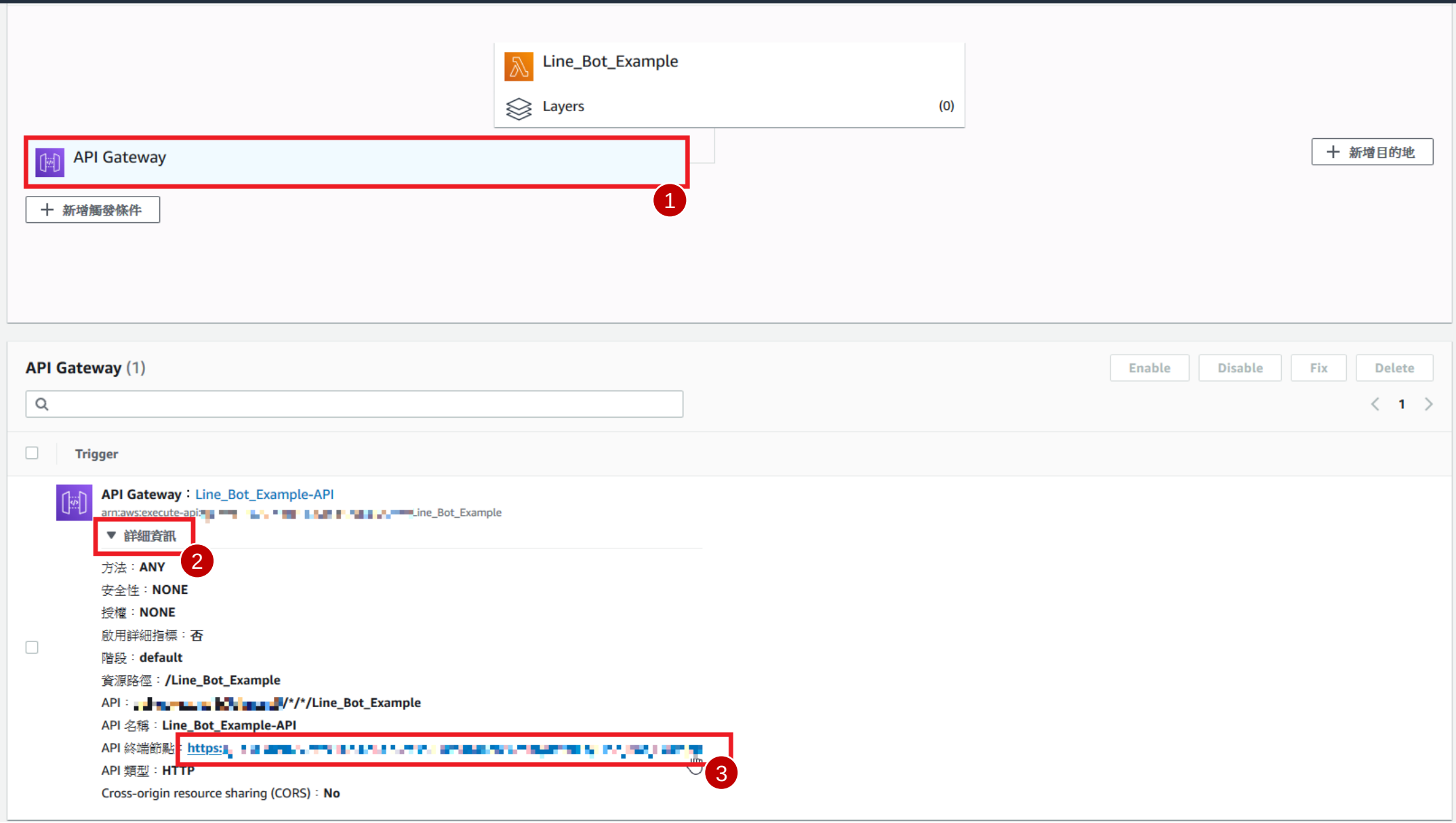
螢幕擷圖參考資訊 – 步驟 1



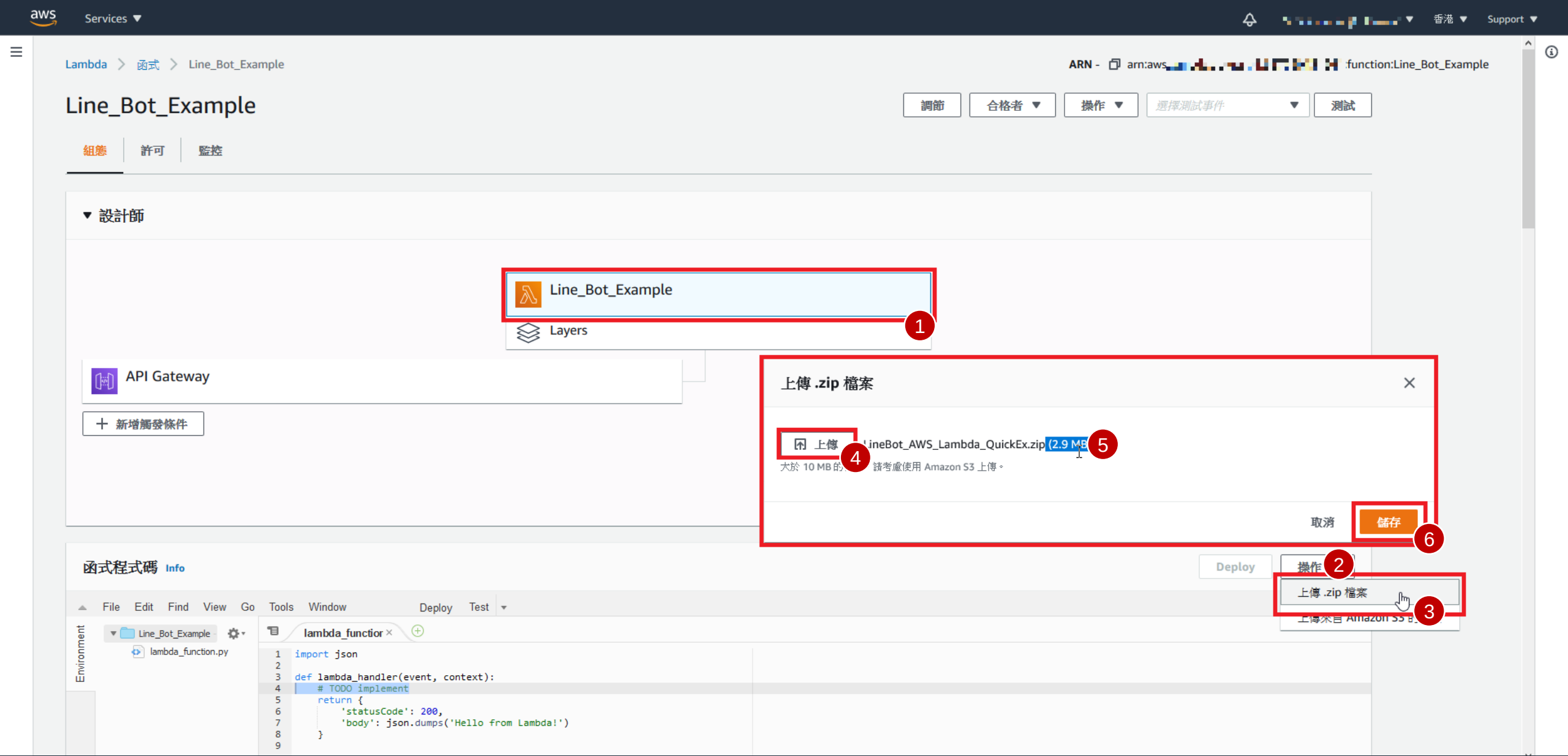
螢幕擷圖參考資訊


aws
Services
香港
Support
Lambda > 函式 > Line_Bot_Example
ARN - arn:aws:lambda:ap-east-1:627750131661:function:Line_Bot_Example
Line_Bot_Example
調節
合格者
操作
選擇測試事件
測試
組態
許可
監控
設計師
Line_Bot_Example
Layers
新增觸發條件
1
觸發組態
選取 觸發條件
API Gateway
api application-services aws serverless
2
AWS IoT
aws devices iot
Alexa Skills Kit
alexa iot
CloudWatch Logs
函式程式碼 Info
File Edit Find
Environment
Line_Bot_Example
lambda_function.py
lambda_functior
import json
def lambda_handler(event, context):
# TODO implement
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!')
}
觸發組態
API Gateway
api application-services aws serverless
Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. Learn more
API
Create a new API or attach an existing one.
Create an API
3
API type
4
HTTP API
Create an HTTP API.
REST API
Create a REST API.
安全性
Configure the security mechanism for your API endpoint.
開啟
5
其他設置
Lambda 會增加 Amazon API Gateway 的必要權限，以便用這個觸發條件叫用您的 Lambda 函式。進一步了解 Lambda 權限模型。
6
取消
新增

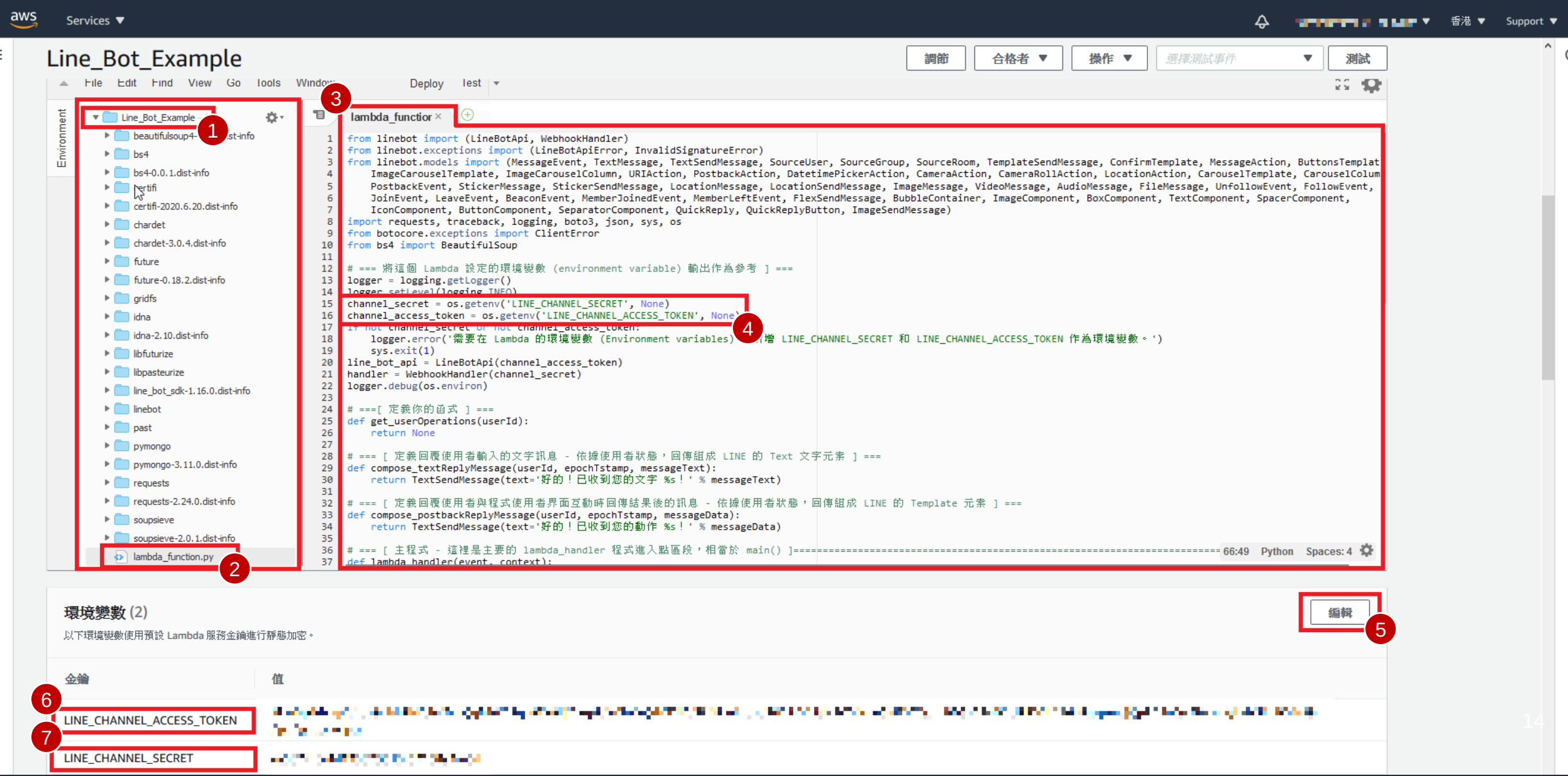
螢幕擷圖參考資訊


Line_Bot_Example
Layers
(0)
API Gateway
1
+ 新增觸發條件
+ 新增目的地
API Gateway (1)
Enable
Disable
Fix
Delete
1
Trigger
API Gateway：Line_Bot_Example-API
arn:aws:execute-api:
ine_Bot_Example
詳細資訊
2
方法：ANY
安全性：NONE
授權：NONE
啟用詳細指標：否
階段：default
資源路徑：/Line_Bot_Example
API：
/*/*/Line_Bot_Example
API 名稱：Line_Bot_Example-API
API 終端節點：https:
API 類型：HTTP
3
Cross-origin resource sharing (CORS)：No

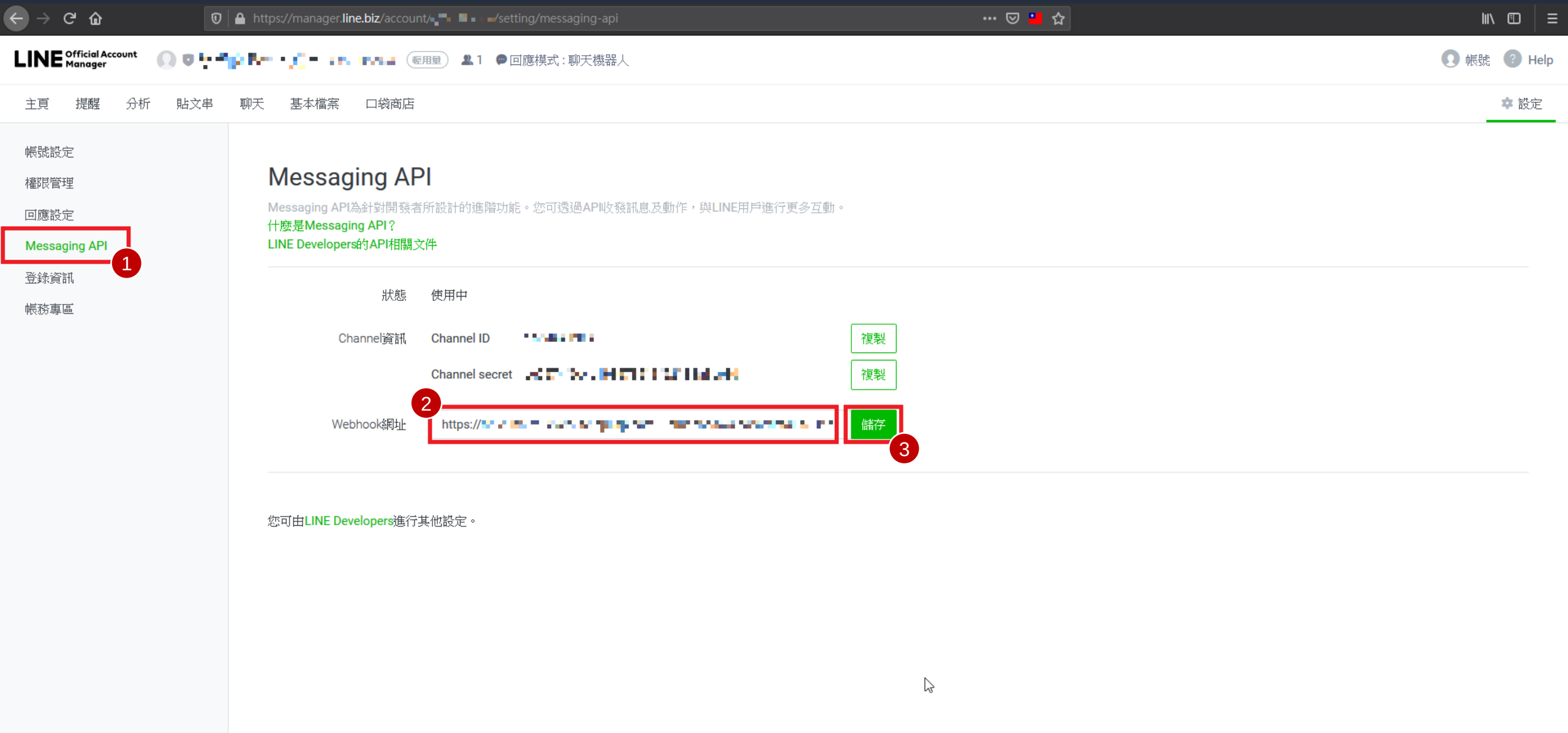
螢幕擷圖參考資訊 – 步驟 2



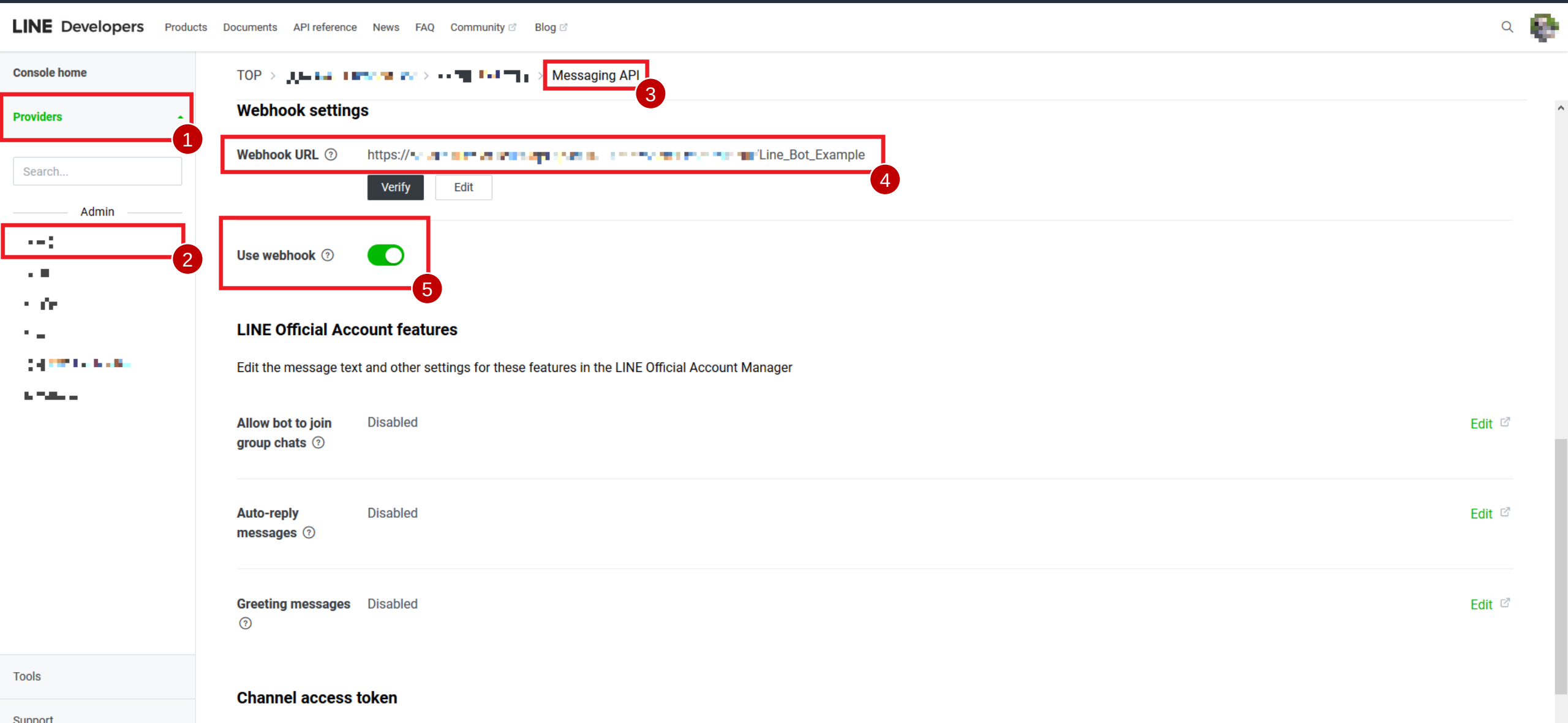
螢幕擷圖參考資訊


Services
香港
Support
Line_Bot_Example
調節
合格者
操作
選擇測試事件
測試
File Edit Find View Go Tools Window
Deploy
Test
Environment
Line_Bot_Example
beautifulsoup4-
bs4
bs4-0.0.1.dist-info
certifi
certifi-2020.6.20.dist-info
chardet
chardet-3.0.4.dist-info
future
future-0.18.2.dist-info
gridfs
idna
idna-2.10.dist-info
libfuturize
libpasteurize
line_bot_sdk-1.16.0.dist-info
linebot
past
pymongo
pymongo-3.11.0.dist-info
requests
requests-2.24.0.dist-info
soupsieve
soupsieve-2.0.1.dist-info
lambda_function.py
lambda_functior
from linebot import (LineBotApi, WebhookHandler)
from linebot.exceptions import (LineBotApiError, InvalidSignatureError)
from linebot.models import (MessageEvent, TextMessage, TextSendMessage, SourceUser, SourceGroup, SourceRoom, TemplateSendMessage, ConfirmTemplate, MessageAction, ButtonsTemplat
ImageCarouselTemplate, ImageCarouselColumn, URIAction, PostbackAction, DatetimePickerAction, CameraAction, CameraRollAction, LocationAction, CarouselTemplate, CarouselColum
PostbackEvent, StickerMessage, StickerSendMessage, LocationMessage, LocationSendMessage, ImageMessage, VideoMessage, AudioMessage, FileMessage, UnfollowEvent, FollowEvent,
JoinEvent, LeaveEvent, BeaconEvent, MemberJoinedEvent, MemberLeftEvent, FlexSendMessage, BubbleContainer, ImageComponent, BoxComponent, TextComponent, SpacerComponent,
IconComponent, ButtonComponent, SeparatorComponent, QuickReply, QuickReplyButton, ImageSendMessage)
import requests, traceback, logging, boto3, json, sys, os
from botocore.exceptions import ClientError
from bs4 import BeautifulSoup
# === 將這個 Lambda 設定的環境變數 (environment variable) 輸出作為參考] ===
logger = logging.getLogger()
logger.setLevel(logging.INFO)
channel_secret = os.getenv('LINE_CHANNEL_SECRET', None)
channel_access_token = os.getenv('LINE_CHANNEL_ACCESS_TOKEN', None)
if not channel_secret or not channel_access_token:
logger.error('需要在 Lambda 的環境變數 (Environment variables) 增增 LINE_CHANNEL_SECRET 和 LINE_CHANNEL_ACCESS_TOKEN 作為環境變數。')
sys.exit(1)
line_bot_api = LineBotApi(channel_access_token)
handler = WebhookHandler(channel_secret)
logger.debug(os.environ)
# ===[定義你的函式] ===
def get_userOperations(userId):
return None
# === [定義回覆使用者輸入的文字訊息 - 依據使用者狀態，回傳組成 LINE 的 Text 文字元素] ===
def compose_textReplyMessage(userId, epochTstamp, messageText):
return TextSendMessage(text='好的！已收到您的文字 %s！' % messageText)
# === [定義回覆使用者與程式使用者界面互動時回傳結果後的訊息 - 依據使用者狀態，回傳組成 LINE 的 Template 元素] ===
def compose_postbackReplyMessage(userId, epochTstamp, messageData):
return TextSendMessage(text='好的！已收到您的動作 %s！' % messageData)
=== [主程式 - 這裡是主要的 lambda_handler 程式進入點區段，相當於 main()]
66:49 Python Spaces: 4
def lambda_handler(event, context):
1
2
3
4
5
環境變數 (2)
以下環境變數使用預設 Lambda 服務金鑰進行靜態加密。
編輯
金鑰
值
6
LINE_CHANNEL_ACCESS_TOKEN
7
LINE_CHANNEL_SECRET

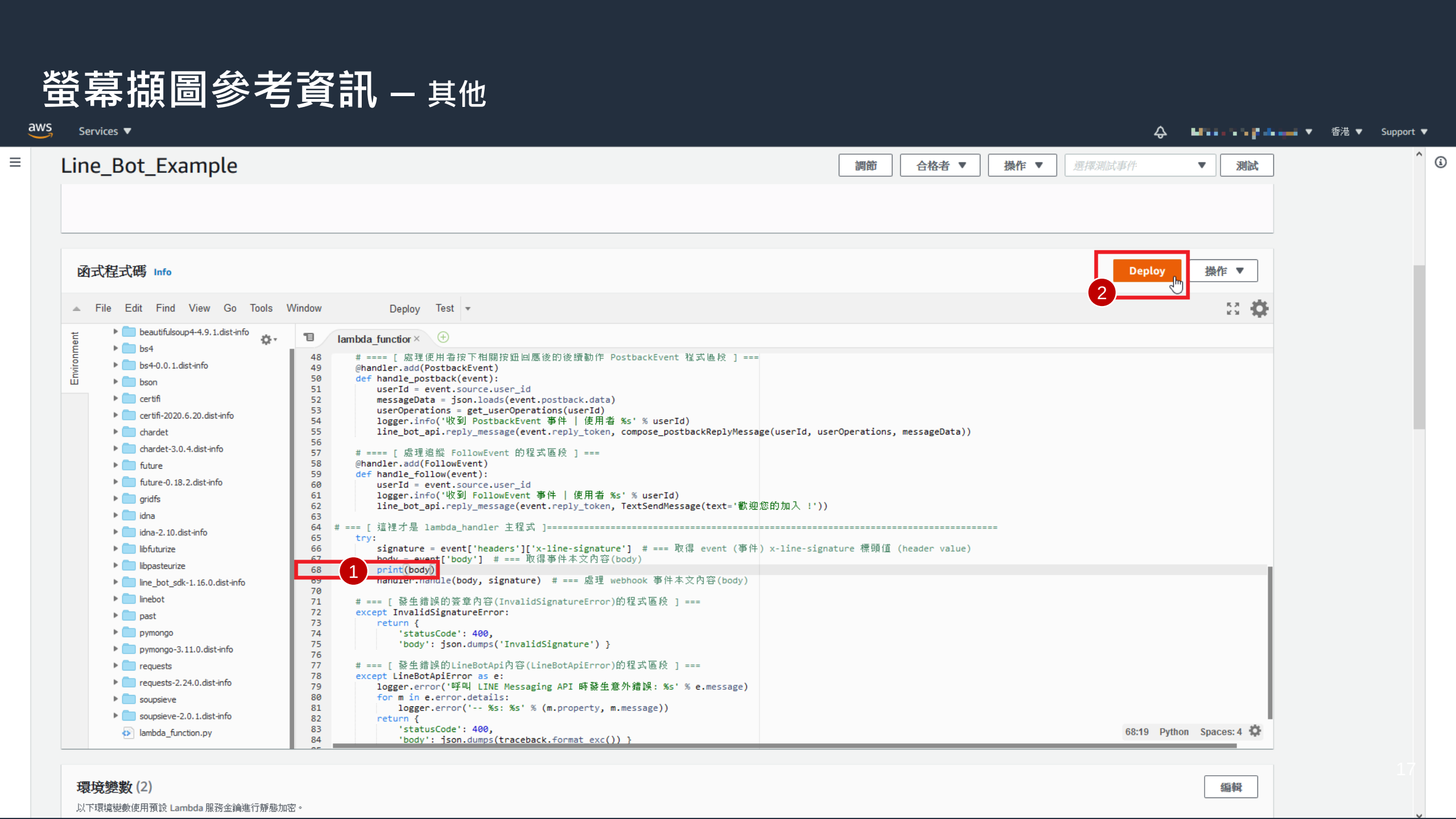
螢幕擷圖參考資訊 – 步驟 3



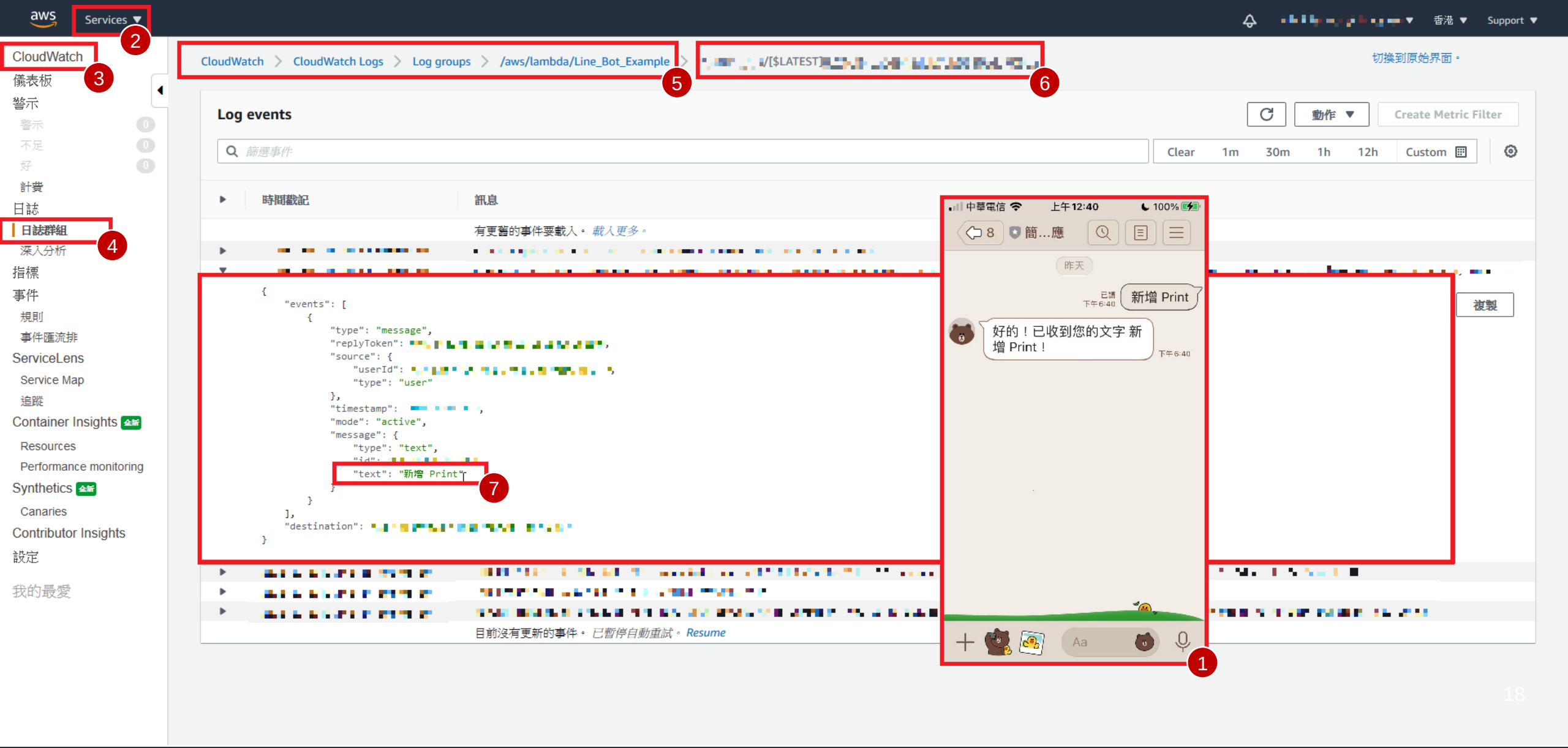
螢幕擷圖參考資訊


LINE Developers
Products
Documents
API reference
News
FAQ
Community
Blog
Console home
Providers
1
Search...
Admin
2
TOP
Messaging API
3
Webhook settings
Webhook URL
https://
Line_Bot_Example
4
Verify
Edit
Use webhook
5
LINE Official Account features
Edit the message text and other settings for these features in the LINE Official Account Manager
Allow bot to join group chats
Disabled
Edit
Auto-reply messages
Disabled
Edit
Greeting messages
Disabled
Edit
Channel access token
Tools
Support

螢幕擷圖參考資訊 – 其他


aws
Services
香港
Support
Line_Bot_Example
調節
合格者
操作
選擇測試事件
測試
函式程式碼 Info
Deploy
操作
2
File Edit Find View Go Tools Window
Deploy Test
Environment
beautifulsoup4-4.9.1.dist-info
bs4
bs4-0.0.1.dist-info
bson
certifi
certifi-2020.6.20.dist-info
chardet
chardet-3.0.4.dist-info
future
future-0.18.2.dist-info
gridfs
idna
idna-2.10.dist-info
libfuturize
libpasteurize
line_bot_sdk-1.16.0.dist-info
linebot
past
pymongo
pymongo-3.11.0.dist-info
requests
requests-2.24.0.dist-info
soupsieve
soupsieve-2.0.1.dist-info
lambda_function.py
lambda_functior
48 # ==== [處理使用者按下相關按鈕回應後的後續動作 PostbackEvent 程式區段] ===
49 @handler.add(PostbackEvent)
50 def handle_postback(event):
51 userId = event.source.user_id
52 messageData = json.loads(event.postback.data)
53 userOperations = get_userOperations(userId)
54 logger.info('收到 PostbackEvent 事件 | 使用者 %s' % userId)
55 line_bot_api.reply_message(event.reply_token, compose_postbackReplyMessage(userId, userOperations, messageData))
56
57 # ==== [處理追縱 FollowEvent 的程式區段] ===
58 @handler.add(FollowEvent)
59 def handle_follow(event):
60 userId = event.source.user_id
61 logger.info('收到 FollowEvent 事件 | 使用者 %s' % userId)
62 line_bot_api.reply_message(event.reply_token, TextSendMessage(text='歡迎您的加入 !'))
63
64 # === [這裡才是 lambda_handler 主程式]==========
65 try:
66 signature = event['headers']['x-line-signature'] # === 取得 event (事件) x-line-signature 標頭值 (header value)
67 body = event['body'] # === 取得事件本文內容(body)
68 print(body)
69 handler.handle(body, signature) # === 處理 webhook 事件本文內容(body)
70
71 # === [發生錯誤的簽章內容(InvalidSignatureError)的程式區段] ===
72 except InvalidSignatureError:
73 return {
74 'statusCode': 400,
75 'body': json.dumps('InvalidSignature') }
76
77 # === [發生錯誤的LineBotApi內容(LineBotApiError)的程式區段] ===
78 except LineBotApiError as e:
79 logger.error('呼叫 LINE Messaging API 時發生意外錯誤: %s' % e.message)
80 for m in e.error.details:
81 logger.error('-- %s: %s' % (m.property, m.message))
82 return {
83 'statusCode': 400,
84 'body': json.dumps(traceback.format_exc()) }
1
68:19 Python Spaces: 4
環境變數 (2)
以下環境變數使用預設 Lambda 服務金鑰進行靜態加密。
編輯

螢幕擷圖參考資訊


aws
Services
2
香港
Support
CloudWatch
3
儀表板
警示
警示
不足
好
計費
日誌
日誌群組
4
深入分析
指標
事件
規則
事件匯流排
ServiceLens
Service Map
追蹤
Container Insights
全新
Resources
Performance monitoring
Synthetics
全新
Canaries
Contributor Insights
設定
我的最愛
CloudWatch > CloudWatch Logs > Log groups > /aws/lambda/Line_Bot_Example
5
/[$LATEST]
6
切換到原始界面。
Log events
動作
Create Metric Filter
篩選事件
Clear
1m
30m
1h
12h
Custom
時間戳記
訊息
有更舊的事件要載入。 載入更多。
{
"events": [
{
"type": "message",
"replyToken":
"source": {
"userId":
"type": "user"
},
"timestamp":
"mode": "active",
"message": {
"type": "text",
"text": "新增 Print"
}
}
],
"destination":
}
7
中華電信
上午12:40
100%
8
簡...應
昨天
新增 Print
好的！已收到您的文字 新增 Print！
下午6:40
Aa
1
複製
目前沒有更新的事件。 已暫停自動重試。 Resume