AI-102 uuu 2021/11/8,9,10,11

doc url: https://wwjd.tw/664k392 (https://wwjd.tw/664k392)

挑戰賽:
https://docs.google.com/spreadsheets/d/1QIKK-DNj9JYAvUDKsERhvVsNzEZ7Kzz6G4qq2kcueu8/edit?usp=sharing (https://docs.google.com/spreadsheets/d/1QIKK-DNj9JYAvUDKsERhvVsNzEZ7Kzz6G4qq2kcueu8/edit?usp=sharing)

貼上圖片換URL:
https://hackmd.io/@twdeveloper/HkSlnf0Ut (https://hackmd.io/@twdeveloper/HkSlnf0Ut)

帳號建立:

MS Acount:
https://account.microsoft.com/account?lang=zh-hk (https://account.microsoft.com/account?lang=zh-hk)

Azure Pass:
https://www.microsoftazurepass.com/ (https://www.microsoftazurepass.com/)

azure portal:
https://portal.azure.com/ (https://portal.azure.com/)

Azure Pass剩餘金額:
https://www.microsoftazuresponsorships.com/Balance (https://www.microsoftazuresponsorships.com/Balance)

azure free trial: (需要信用卡)
https://azure.microsoft.com/zh-tw/free/ (https://azure.microsoft.com/zh-tw/free/)

~~VS essentials(VS2019替代方案):~~
~~https://visualstudio.microsoft.com/zh-hant/dev-essentials/~~ (~~https://visualstudio.microsoft.com/zh-hant/dev-essentials/~~)

開發工具

VS Code:
https://code.visualstudio.com/ (https://code.visualstudio.com/)

VS 2019 community(good enough):
https://visualstudio.microsoft.com/zh-hant/downloads/ (https://visualstudio.microsoft.com/zh-hant/downloads/)

Postman:
https://www.postman.com/ (https://www.postman.com/)

相關SDK與工具(先安裝VS2019):

Bot Framework v4 SDK Templates for Visual Studio:
https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4 (https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4)

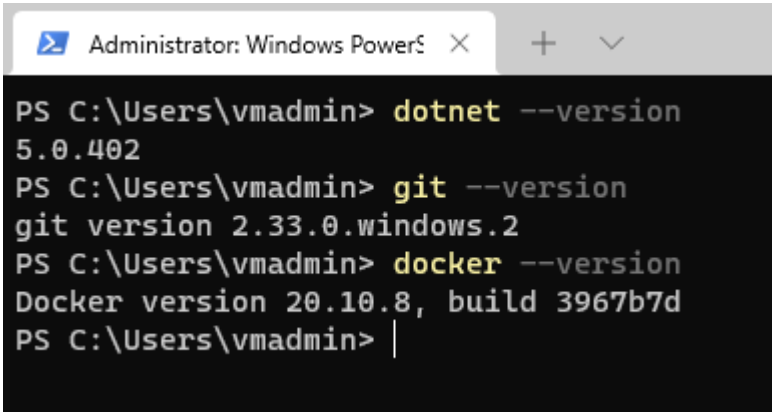
BotFramework-Emulator:
https://github.com/microsoft/BotFramework-Emulator/releases/download/v4.11.0/BotFramework-Emulator-4.11.0-windows-setup.exe (https://github.com/microsoft/BotFramework-Emulator/releases/download/v4.11.0/BotFramework-Emulator-4.11.0-windows-setup.exe)

目標:
環境準備

步驟:
1. 建立MS Account
2. 申請Azure Pass
3. 進入Azure Portal
4. 準備VM環境
    1. Create win10-20H2 VM on D4sV3
    2. VS2019(升級到最新版)
    3. Docker Running
    4. Bot SDK https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4
    5. Bot Emulator

check list

```
dotnet --version
git --version
docker --version
```



- [x] VS 2019
- [x] VS Code
- [x] docker desktop
- [x] dotnet 3.1+
- [x] git
- [x] Azure Portal and subscription

Lab 1: 建立與使用Computer Vision API

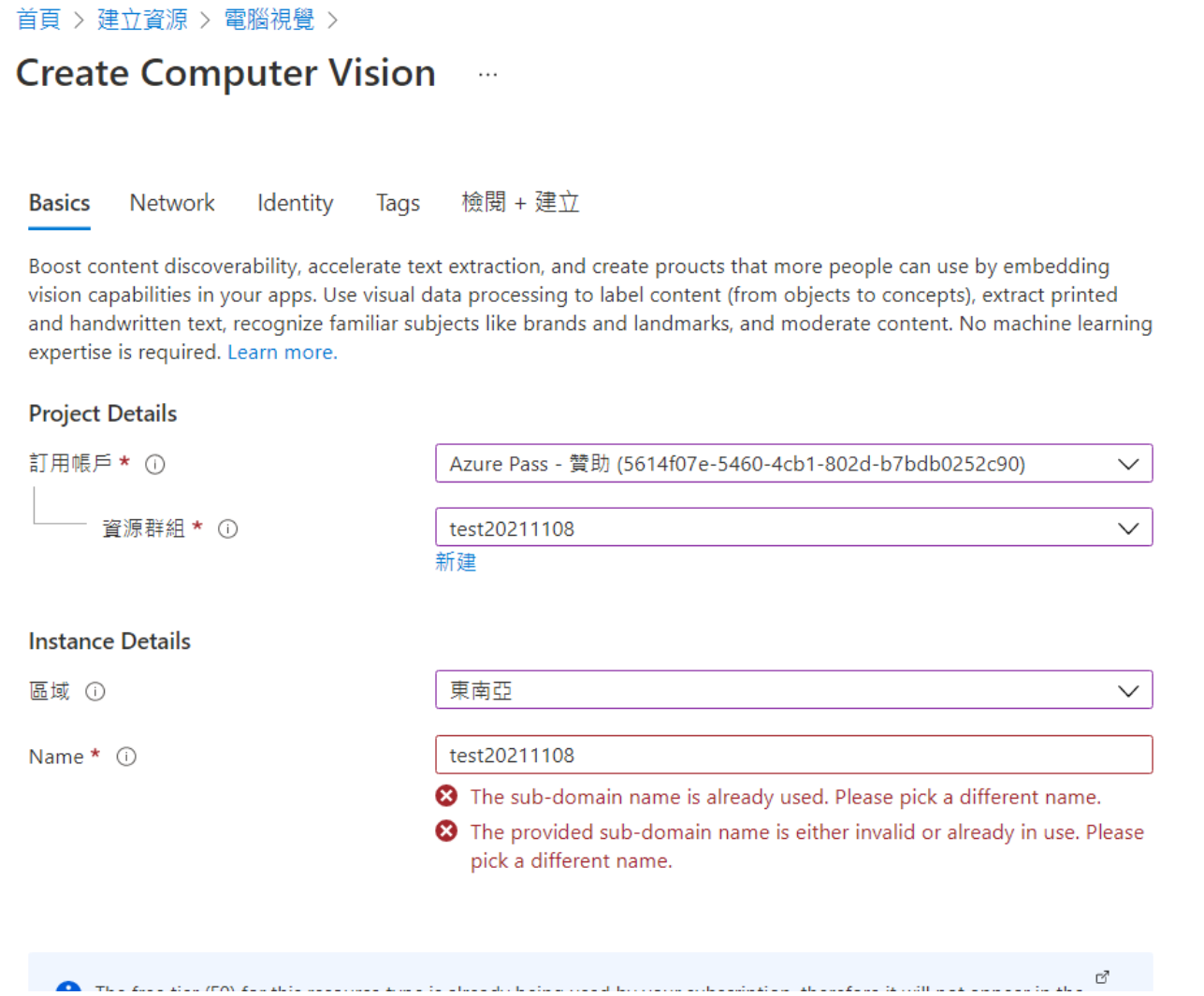
步驟:

1. 建立Computer Vision (電腦視覺)
2. 取得金鑰與端點
3. 查找文件 https://westcentralus.dev.cognitive.microsoft.com/docs/services/computer-vision-v3-1-ga/operations/5d986960601faab4bf452005
4. 使用 v3.1 API
5. 使用 postman呼叫API
    1. 辨識圖片
    2. 辨識人臉
6. 使用程式碼呼叫

reference:

Computer Vision Video:
https://youtu.be/rU4I3nRGaxw (https://youtu.be/rU4I3nRGaxw)

建立資源:



可以用的圖片(JSON):

```
{"url":"https://i.imgur.com/o8Jyv9d.jpg"}
```

借你用的金鑰 & 端點:
fe2c80095cbf4652885bfac42deec3c7
https://test20211108.cognitiveservices.azure.com/
(https://test20211108.cognitiveservices.azure.com/)

Lab 2: 使用cognitive services

步驟:

1. 建立通用 cognitive Services，取得端點與金鑰匙
2. Git clone https://github.com/MicrosoftLearning/AI-102-AIEngineer
   (https://github.com/MicrosoftLearning/AI-102-AIEngineer)
3. dotnet add package Azure.AI.TextAnalytics
4. dotnet build
5. Code . <–開啟VS code

6. 修改 appsettings.json
7.
```
{
    "CognitiveServicesEndpoint": "YOUR_COGNITIVE_SERVICES_ENDPOINT",
    "CognitiveServiceKey": "YOUR_COGNITIVE_SERVICES_KEY"
}
```
8. dotnet run

```
PS D:\ai102\AI-102-AIEngineer\01-getting-started\C-Sharp\sdk-client> dotnet run

Enter some text ('quit' to stop)
今天開始變冷
Language: Chinese_Traditional

Enter some text ('quit' to stop)
```

ref: 課本lab: https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/01-get-started-cognitive-services.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/01-get-started-cognitive-services.html)

command ref:

```
#建立資料夾
md 資料夾名稱
#git版控指，複製github雲端檔案
git clone https://github.com/MicrosoftLearning/AI-102-AIEngineer
#進入資料夾
cd 資料夾名稱
#開啟VS code
Code .
#安裝nuget套件
dotnet add package Azure.AI.TextAnalytics
#build程式
dotnet build
#執行程式
dotnet run
```

Lab 3: 使用Container

步驟:

1. 確認docker is running
2. 建立並取得Text Analysis(或cognitive services)金鑰與端點
3. 執行docker run 指令

```
docker run --rm -it -p 127.0.0.1:5000:5000 --memory 4g --cpus 1 mcr.micros
```

1. 執行postman 嘗試呼叫local端點

   ====================

   POST /text/analytics/v3.0/languages HTTP/1.1

   Host: 127.0.0.1:5000

   Ocp-Apim-Subscription-Key: 798ff63e8c494c029dc2280dd5766347

   Content-Type: application/json

   Content-Length: 458

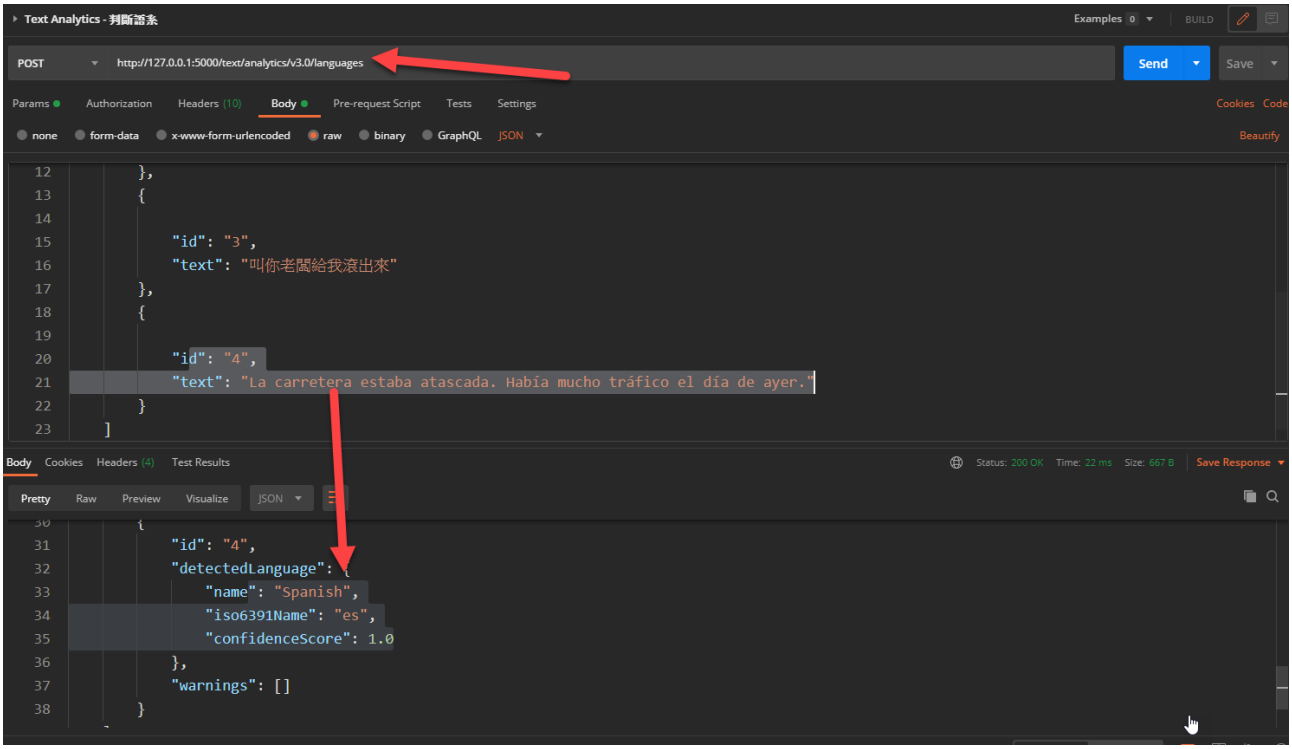
```
{
    "documents": [
        {

            "id": "1",
            "text": "你他媽的這什麼鬼？"
        },
        {

            "id": "2",
            "text": "早安唷，你老闆在嗎？"
        },
        {

            "id": "3",
            "text": "叫你老闆給我滾出來"
        },
        {

            "id": "4",
            "text": "La carretera estaba atascada. Había mucho tráfico el
        }
    ]
}
```



ref:
https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/04-use-a-container.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/04-use-a-container.html)

Lab 4:

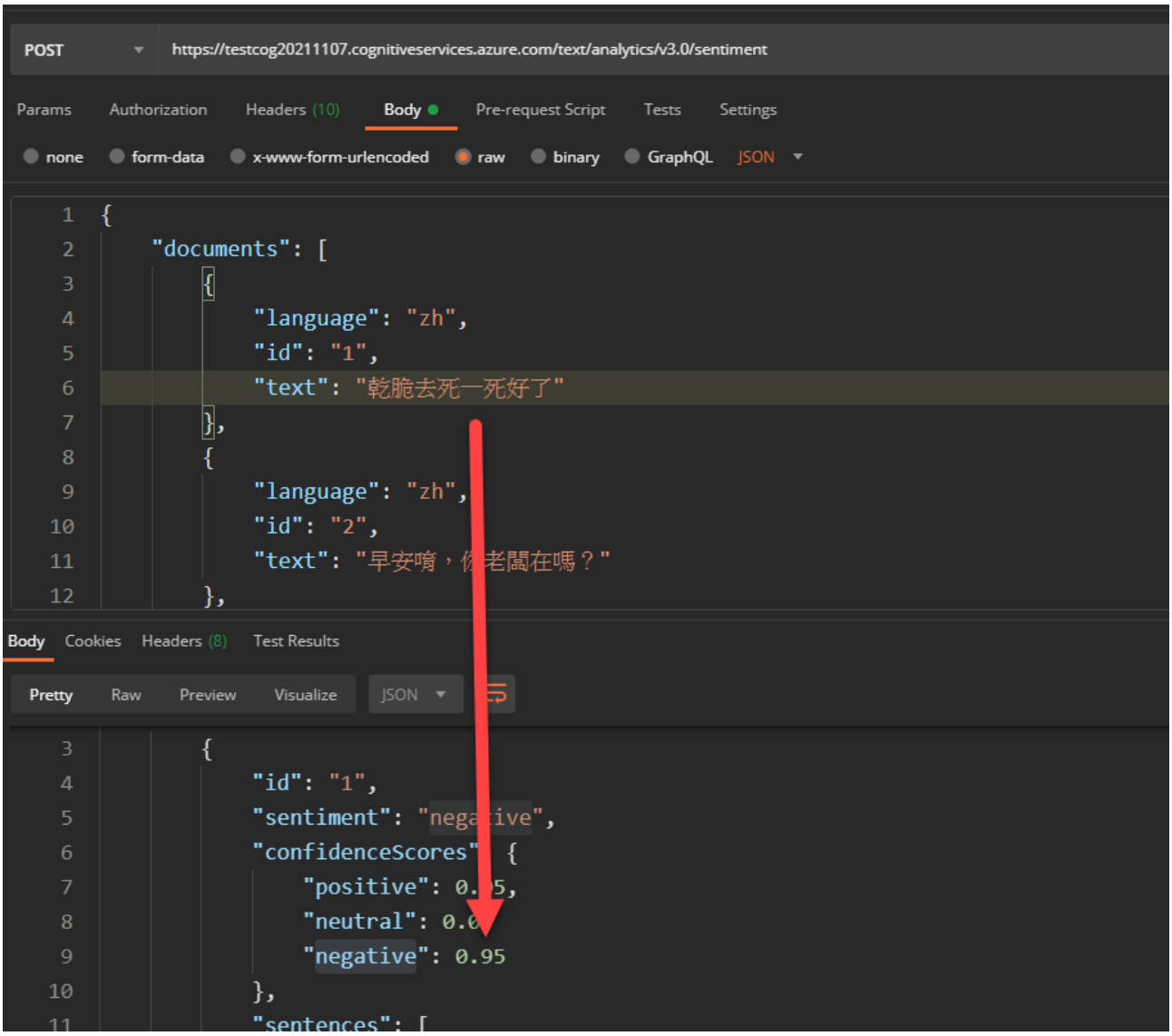
目標:

分析文字情緒

步驟:

1. 建立 Text Analytics服務
2. 取得 endpoint & Key
3. 查考文件
   https://westus2.dev.cognitive.microsoft.com/docs/services/TextAnalytics-v3-1/operations/Sentiment
4. 使用postman
5. 判斷文字情緒
   ex.
   乾脆去玩一玩好了
   vs
   乾脆去死一死好了

result:



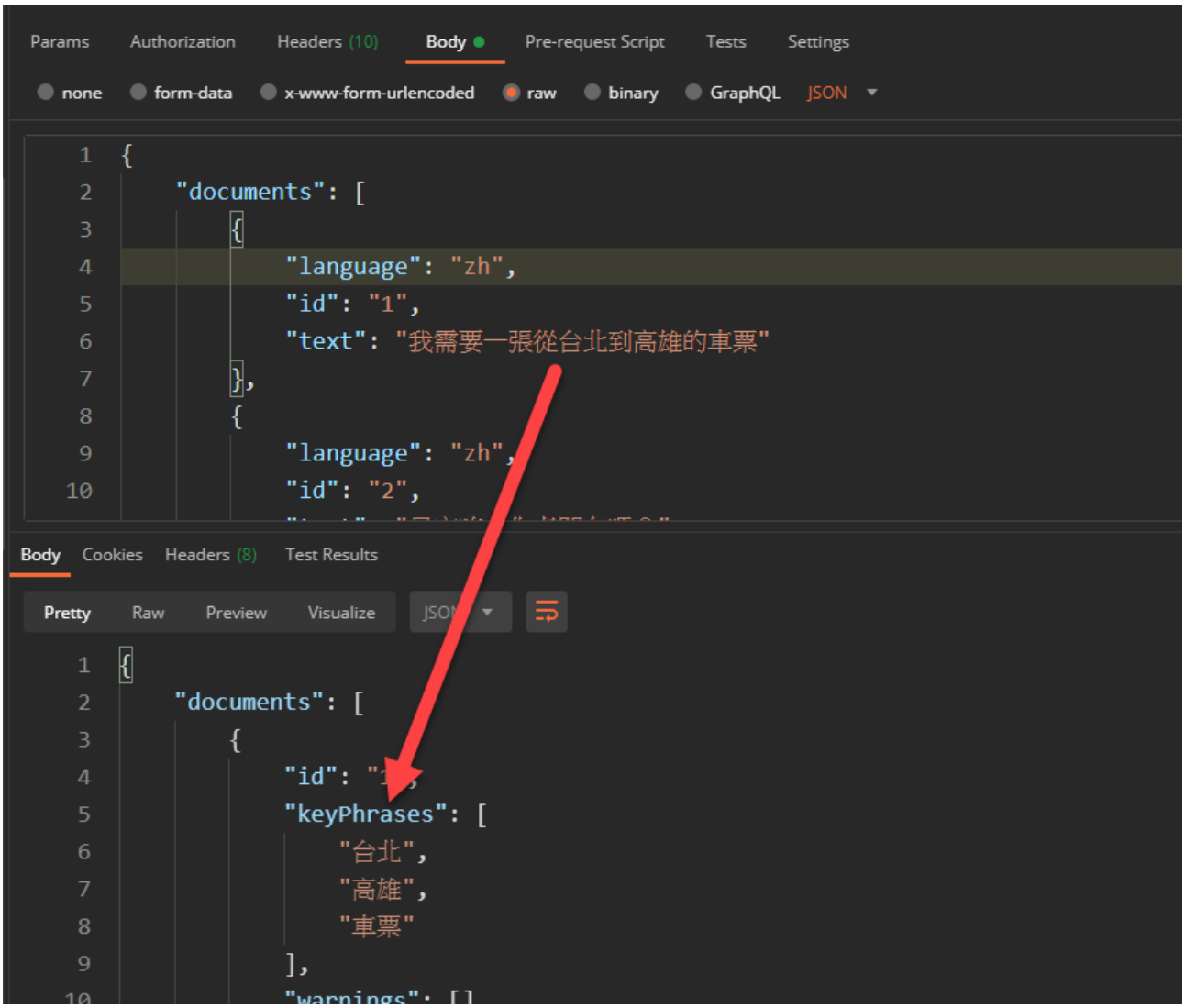
目標:

擷取關鍵字詞

步驟:

1. 建立 Text Analytics服務
2. 取得 endpoint & Key
3. 查考文件
   https://westus2.dev.cognitive.microsoft.com/docs/services/TextAnalytics-v3-1/operations/KeyPhrases
4. 使用postman
5. 判斷一段你想分析的話，看是否抓取到關鍵字詞?
   例:我需要一張從臺北到高雄的車票

result:



參考JSON:

```
{
    "documents": [
        {

            "id": "1",
            "text": "我需要一張從臺北到高雄的車票"
        },
        {

            "id": "2",
            "text": "早安唷，你老闆在嗎？"
        },
        {

            "id": "3",
            "text": "叫你老闆給我滾出來"
        },
        {

            "id": "4",
            "text": "La carretera estaba atascada. Había mucho tráfico el
        }
    ]
}
```

課本lab:
https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/05-analyze-text.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/05-analyze-text.html)

Lab 5:

目標:

建立應用程式來 擷取關鍵字詞 或分析文字情緒

步驟:

1. 參考rest api或SDK的code
2. 將上述兩個Lab擇一進行改寫
3. 使用console App 來呼叫API (可用rest or SDK)


```
#建立new console app
dotnet new console
#開啟 VS Code
code .
#裝套件
dotnet add package Azure.AI.TextAnalytics
```

sample code:

```
using System;
using Azure;
using Microsoft.Extensions.Configuration;
using System.Text;
using Azure.AI.TextAnalytics;

namespace textananysis
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.InputEncoding = Encoding.Unicode;
            Console.OutputEncoding = Encoding.Unicode;

            Console.Write("請輸入句子:");
            var text = Console.ReadLine();
            var Language = TextAnalytics.GetLanguage(text);
            Console.WriteLine("Get Language:" + Language);
            var ret = TextAnalytics.GetSentiment(text);
            Console.WriteLine("Get Sentiment:" + ret);
            var KeyPhrases = TextAnalytics.ExtractKeyPhrases(text);
            foreach (var item in KeyPhrases.Value)
            {
                Console.WriteLine("KeyPhrases:" + item);
            }
        }
    }

    public class TextAnalytics
    {
        const string cogSvcKey = "439c1c08a372445cb9b417c6d3d291d3";
        const string cogSvcEndpoint = "https://mycog20211108.cognitiveserv

        public static string GetLanguage(string text)
        {

            // Create client using endpoint and key
            AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
            Uri endpoint = new Uri(cogSvcEndpoint);
            var client = new TextAnalyticsClient(endpoint, credentials);

            // Call the service to get the detected language
            DetectedLanguage detectedLanguage = client.DetectLanguage(text
            return (detectedLanguage.Name);

        }

        public static string GetSentiment(string text)
        {
            // Create client using endpoint and key
            AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
            Uri endpoint = new Uri(cogSvcEndpoint);
            var client = new TextAnalyticsClient(endpoint, credentials);
```

```
55
56              // Call the service to get the detected language
57              var result = client.AnalyzeSentiment(text, "zh");
58              return (result.Value.Sentiment.ToString());
59          }
60
61          public static Response<KeyPhraseCollection> ExtractKeyPhrases(stri
62          {
63
64              // Create client using endpoint and key
65              AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
66              Uri endpoint = new Uri(cogSvcEndpoint);
67              var client = new TextAnalyticsClient(endpoint, credentials);
68
69              // Call the service to get the detected language
70              var KeyPhrases = client.ExtractKeyPhrases(text, "zh");
71              return (KeyPhrases);
72          }
73      }
74  }
```

Translator API

Lab 6 : 翻譯文字

目標:

翻譯文字

步驟:

1. 建立 Translator 服務
2. 取得 Key
3. 查考文件
   https://docs.microsoft.com/zh-tw/azure/cognitive-services/translator/reference/v3-0-translate
4. 使用postman
5. 翻譯文字
   "嗨，大家今天過的好嗎？“
   "端午節快到了，我想吃肉粽 “
   "雙十國慶快到了，我想要出國玩"

sample code(http)

```
POST /translate?api-version=3.0&to=ja&to=en&to=ko HTTP/1.1
Host: api.cognitive.microsofttranslator.com
Ocp-Apim-Subscription-Key: 428b68baaa044d1a9ef195a52a374326
Ocp-Apim-Subscription-Region: eastasia
Content-Type: application/json
Content-Length: 92

[
  {"Text":"嗨，大家今天過的好嗎?"},
  {"Text":"端午節快到了，我想吃肉粽"},
  {"Text":"雙十國慶快到了，我想要出國玩"}
]
```

Lab 7：找出文字的音譯(發音)

目標:

找出文字的音譯(發音)

步驟:

1. 建立 Translator 服務
2. 取得 Key
3. 查考文件
   https://docs.microsoft.com/zh-tw/azure/cognitive-services/translator/reference/v3-0-transliterate
4. 使用postman
5. 翻譯文字

```
[
    { 'Text' : 'こんにちは'  }
]
```

Speech

speech studio

https://speech.microsoft.com/ (https://speech.microsoft.com/)

Lab 8: speaking clock (語音報時)

```
PS D:\test\myspeechclock\AI-102-speaking-clock-final> dotnet run
Ready to use speech service in eastasia
What time is it?
The time is 10:58
中原標準時間, 10:58
PS D:\test\myspeechclock\AI-102-speaking-clock-final>
```

課本lab:

https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/07-speech.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/07-speech.html)

支援的語音:

https://github.com/MicrosoftDocs/azure-docs.zh-tw/blob/master/articles/cognitive-services/Speech-Service/language-support.md (https://github.com/MicrosoftDocs/azure-docs.zh-tw/blob/master/articles/cognitive-services/Speech-Service/language-support.md)

final code:
https://github.com/isdaviddong/AI-102-speaking-clock-final.git
(https://github.com/isdaviddong/AI-102-speaking-clock-final.git)

LUIS

Lab 9: 建立LUIS帳號與服務，建立可接受訂餐的App

1. 在Azure Portal建立LUIS服務
2. 可選擇F0 pricing tier
3. 建議建立新的resource group
4. 完成後，進入 http://luis.ai (http://luis.ai) , 以相同的MSA登入
5. 建立+New App
6. 建立 intent [點餐行為]
7. 建立 [飲料名稱]、[餐點名稱]等entities
8. 填入[點餐行為]例句、訓練

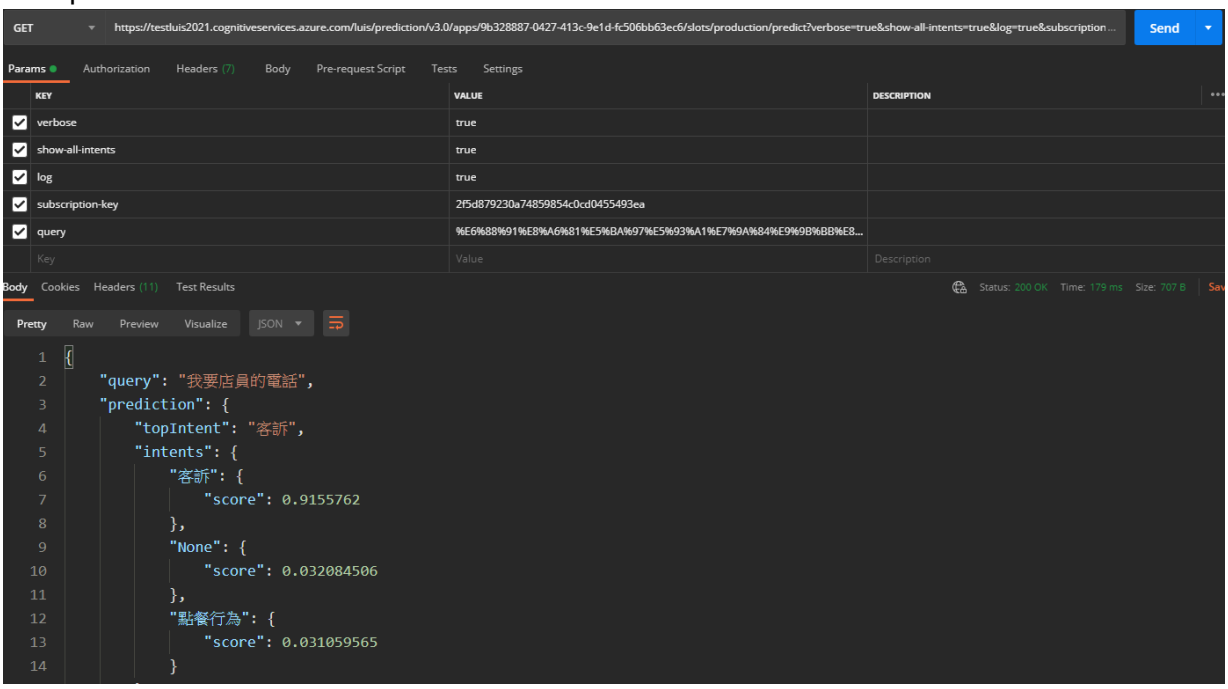
例句:
我要點一份燒餅油條
麻煩你我需要一份蛋餅
給我來個大亨堡
三明治帶走
今天請幫我來一個飯糰
今晚我想來點...巷口的鹹酥雞配薯條

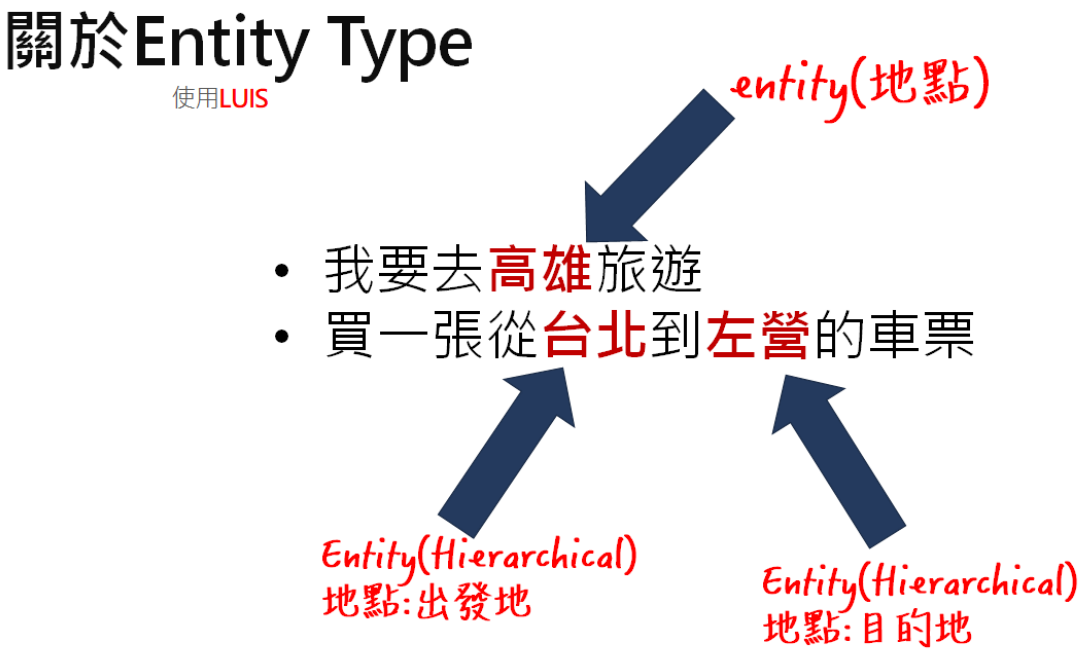
Lab 10: 建立可區分訂餐 vs 客訴的LUIS App

步驟:

1. 建立intent [客訴]
2. 填入[客訴]例句、訓練
3. 透過測試功能確認model是否可順利辨識[客訴]或[點餐]
4. 發佈model

5. 使用postman呼叫測試 <-- 截圖





Lab 11: 透過C# SDK呼叫LUIS

sample code:
https://github.com/isdaviddong/CallLuisSdk30WithDotNetCore31ConsoleApp/blob/main/Program.cs
(https://github.com/isdaviddong/CallLuisSdk30WithDotNetCore31ConsoleApp/blob/main/Program.cs)

```
#create folder
md testluisapp
#into fondler
cd testluisapp
#create console app
dotnet new console
#add nuget pacakge
dotnet add package Microsoft.Azure.CognitiveServices.Language.LUIS.Runtime
#run this app
dotnet run
```

課本:
https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/10-language-understanding-client.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/10-language-understanding-client.html)

LUIS Container:

http://studyhost.blogspot.com/2021/01/luis.html
(http://studyhost.blogspot.com/2021/01/luis.html)

QnA Maker

步驟:

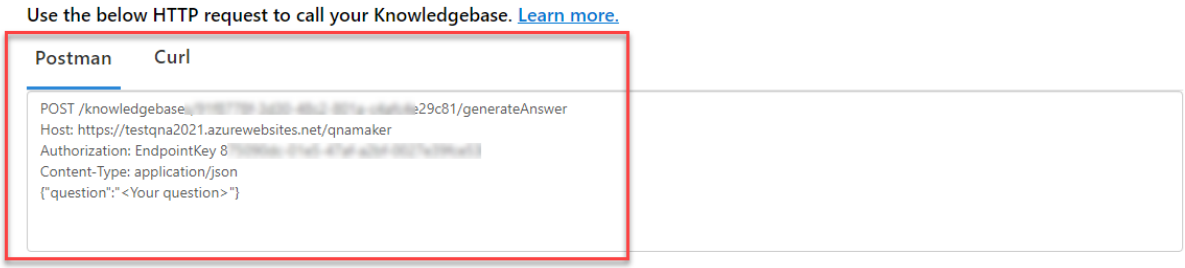
1. 登入Azure Portal
2. 建立QnA Maker
3. 服務選擇S0
4. 索引選擇S(50 indexes) +
5. 東亞(EastAsia)

步驟:

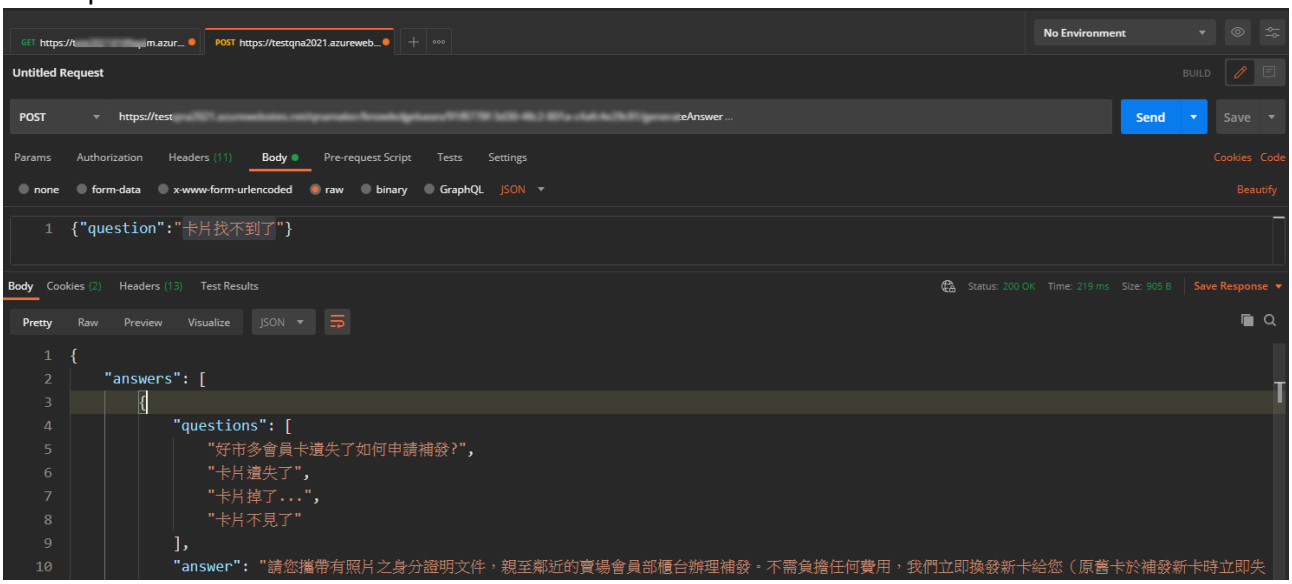
1. 選擇要匯入的PDF或word，或是網頁內容
   ex. https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA (https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA)
2. 亦可手動輸入Q/A
3. 在測試功能中測試(test)
4. 依照需要進行修正，例如...
   1. 何時該我接種?
   2. 打疫苗是否危險?

在postman中呼叫qna maker

A> 先發佈服務



B> 在postman中呼叫



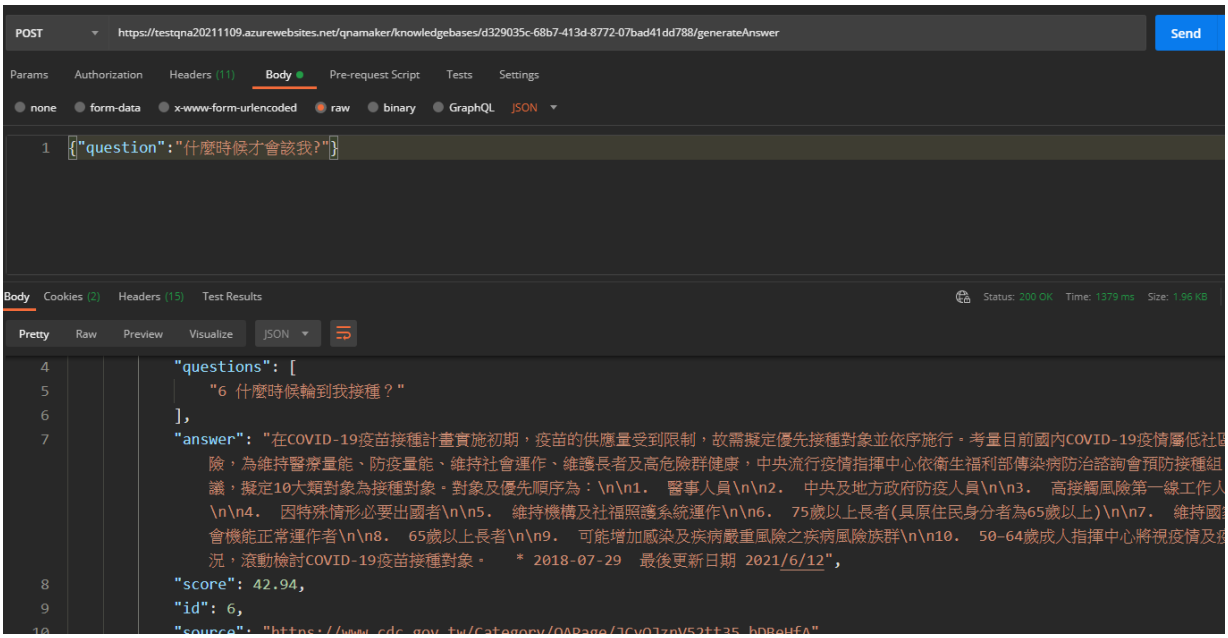

```
curl --location --request POST 'https://tXXXXXXX21.azurewebsites.net/qnama
--header 'Authorization: EndpointKey 875090dc-XXXX-XXXX-a2bf-0027e39fce53'
--header 'Content-Type: application/json' \
--data-raw '{"question":"打疫苗是否危險?"}'
```

Lab 12: 建立並使用QNA Maker幫疾管局製作客服機器人

步驟:

1. 開啟Azure Portal , 建立QnA Maker服務
   https://portal.azure.com/#create/Microsoft.CognitiveServicesQnAMaker (https://portal.azure.com/#create/Microsoft.CognitiveServicesQnAMaker)
2. QnA Maker Console (http://qnamaker.ai (http://qnamaker.ai))
3. 建立KM
   https://www.qnamaker.ai/Create (https://www.qnamaker.ai/Create)
4. 匯入QA
5. 發佈成為服務
   https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA (https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA)
6. 使用postman呼叫 <–截圖

```
curl --location --request POST 'https://tXXXXXXX21.azurewebsites.net/c
--header 'Authorization: EndpointKey 875090dc-XXXX-XXXX-a2bf-0027e39fc
--header 'Content-Type: application/json' \
--data-raw '{"question":"打疫苗是否危險?"}'
```



Create LINE Bot in developer portal

https://youtu.be/gNOYoLOFzpM (https://youtu.be/gNOYoLOFzpM)

透過 .net core 建立LINE Bot(如何建立可Echo的基本LINE Bot)

step by setp hand-on lab:
https://github.com/isdaviddong/HOL-LineBotSDK/blob/master/webhook/Lab 11：如何建立可Echo的 LINE Bot.md (https://github.com/isdaviddong/HOL-LineBotSDK/blob/master/webhook/Lab%2011%20:%20%E5%A6%82%E4%BD%95%E5%BB%BA%E7%AB%8B%E5%8F%AFEcho%E7%9A%84%20LINE%20Bot.md)

```
# script to create .net core web api project
md folder
cd folder
dotnet new webapi
dotnet add package linebotsdk
dotnet new --install isRock.Template.LineWebHook::1.0.12
code .
dotnet new linewebhook
dotnet run
```

ngrok 下載位置:
https://ngrok.com/download (https://ngrok.com/download)

ngrok http 5000 -host-header="localhost:5000"

Ngrok使用說明(video):
https://www.youtube.com/watch?v=0FAEDttEc90&feature=youtu.be
(https://www.youtube.com/watch?v=0FAEDttEc90&feature=youtu.be)

目標:
透過QnA Maker建立Web App Bot
步驟:

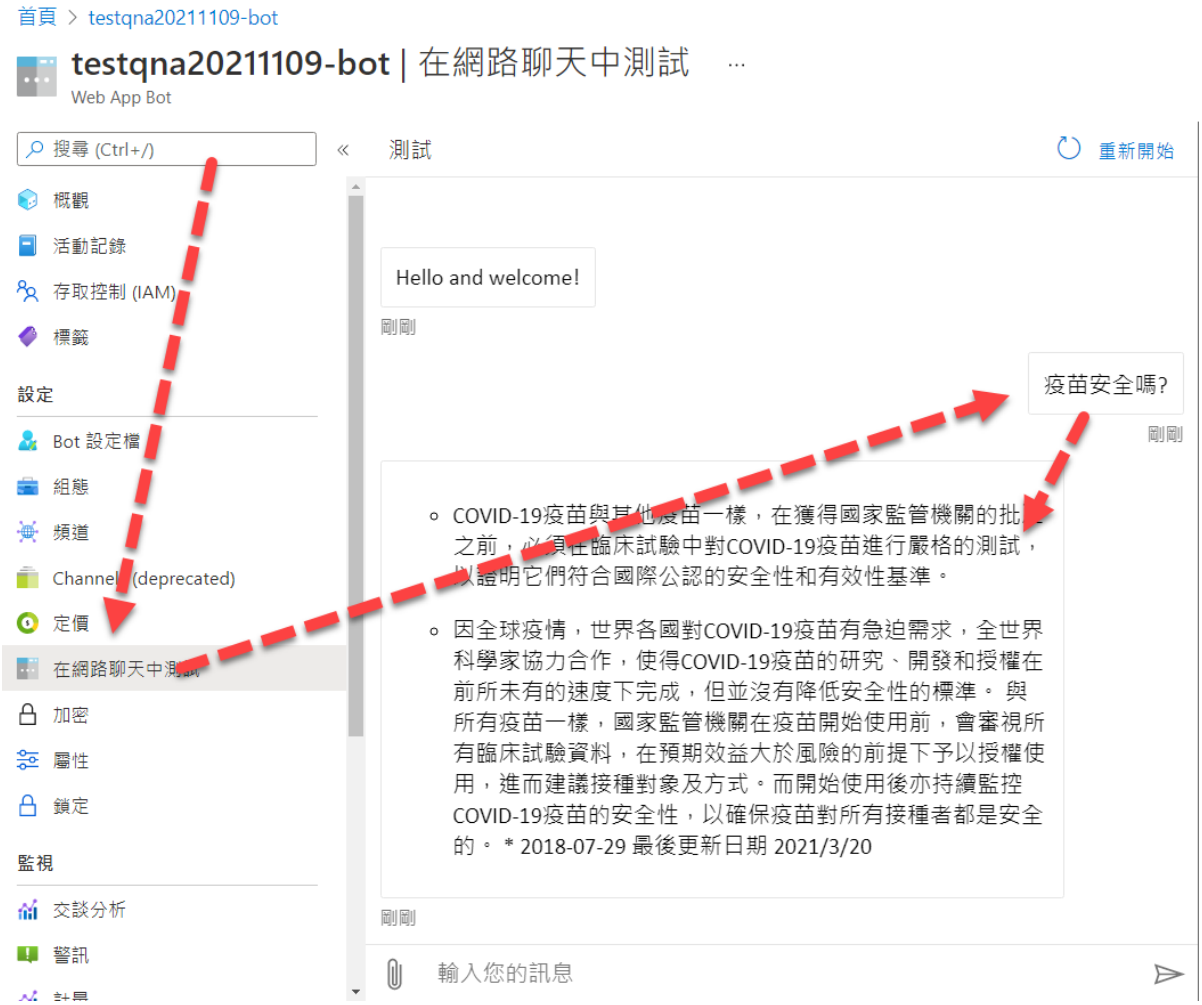
1. 開啟QnA Maker Console
2. 發佈QnA Maker
3. 點選Create Bot
4. 依照流程建立
5. 在Web中測試Web App Bot
6. 透過頻道串接至LINE Bot

Install Bot Framework Composer:
https://docs.microsoft.com/en-us/composer/install-composer?tabs=windows
(https://docs.microsoft.com/en-us/composer/install-composer?tabs=windows)

Lab 13: 透過QnA Maker建立Web App Bot

步驟:

1. 開啟QnA Maker Console
2. 發佈QnA Maker
3. 點選Create Bot
4. 依照流程建立
5. 在Web中測試Web App Bot <-- 截圖



課本 lab ref:

https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/12-qna-maker.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/12-qna-maker.html)

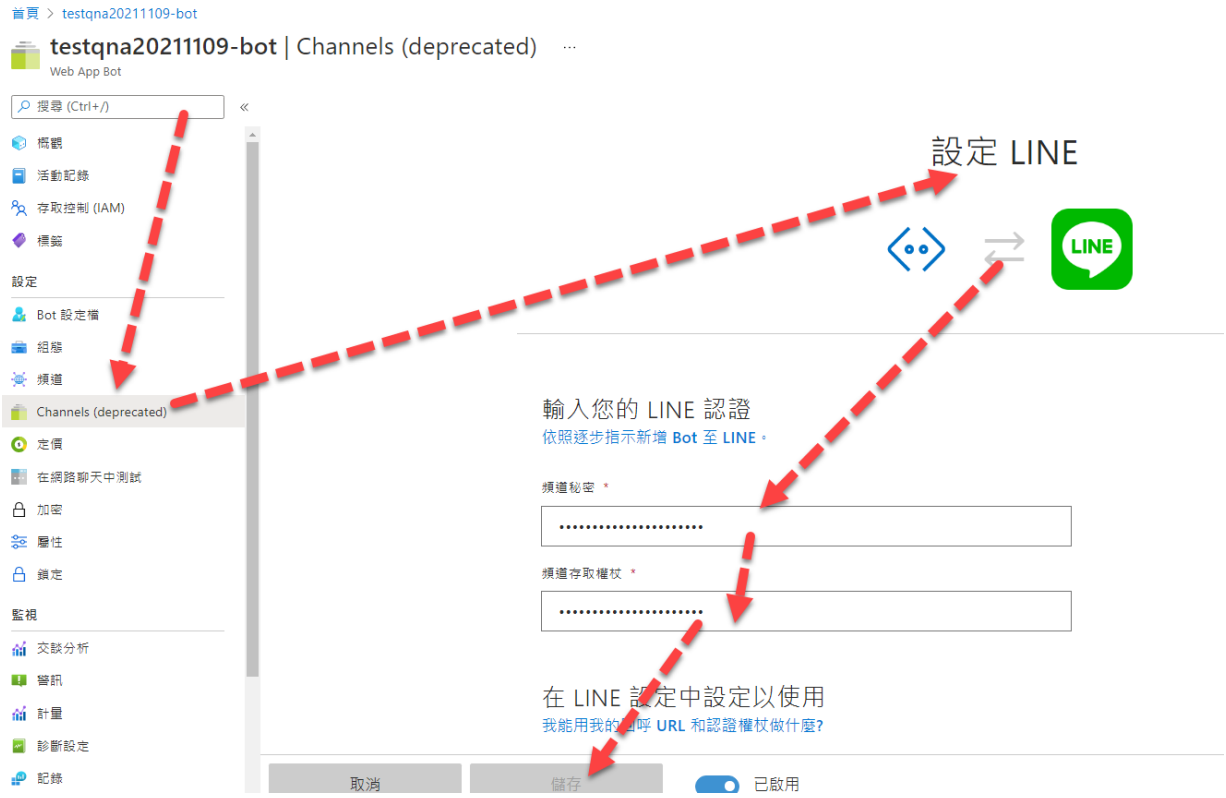
建立第一支LINE Bot (2020版)

https://youtu.be/gNOYoLOFzpM (https://youtu.be/gNOYoLOFzpM)

Lab: 使用LINE channel 串接 QnA Bot

1. 開啟 Web App Bot

2. 設定channel



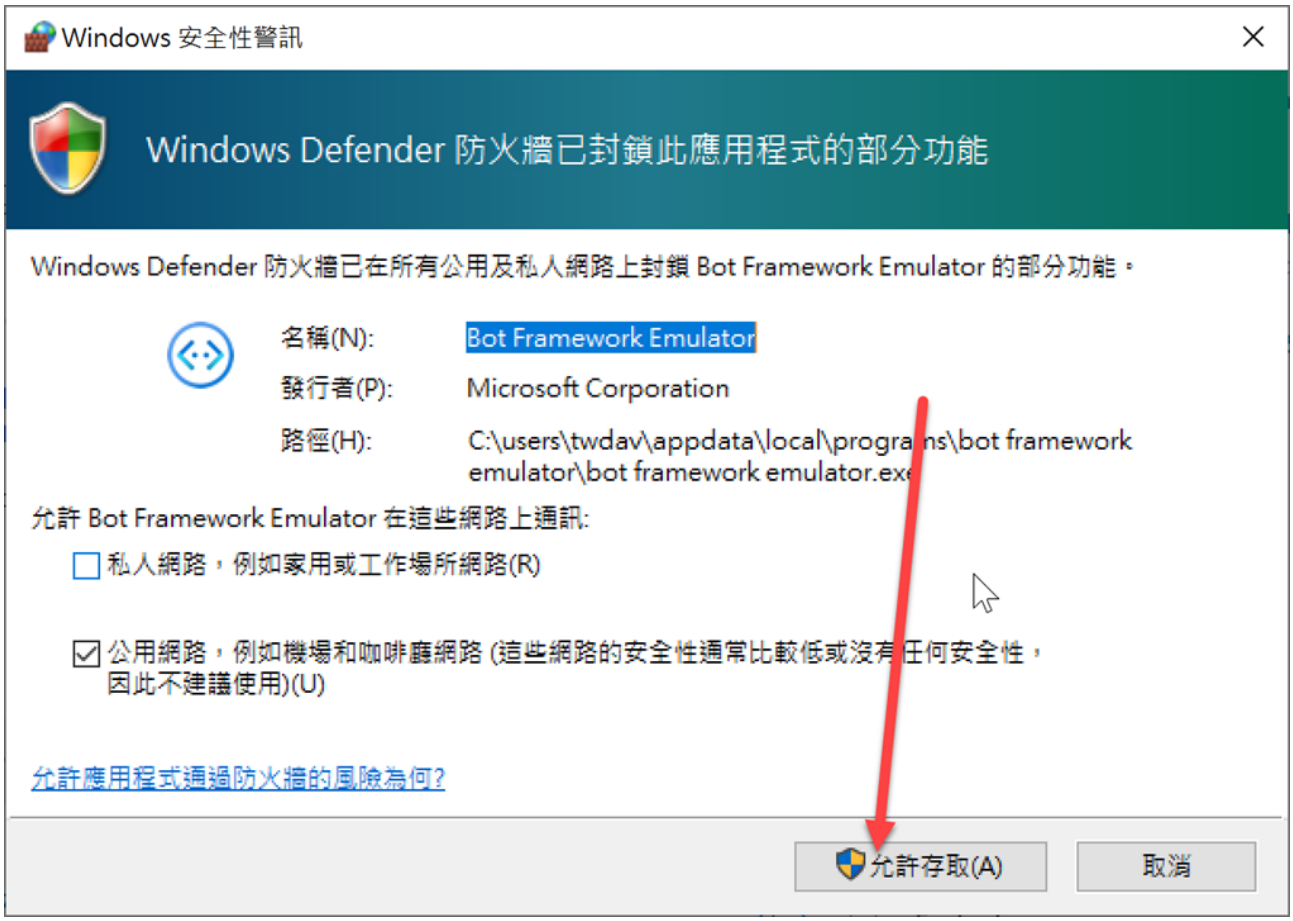
3. 取得WebHook
4. 開啟 LINE 後台
5. 設定從 Azure 取得的 LINE WebHook

Bot Framework

環境準備:

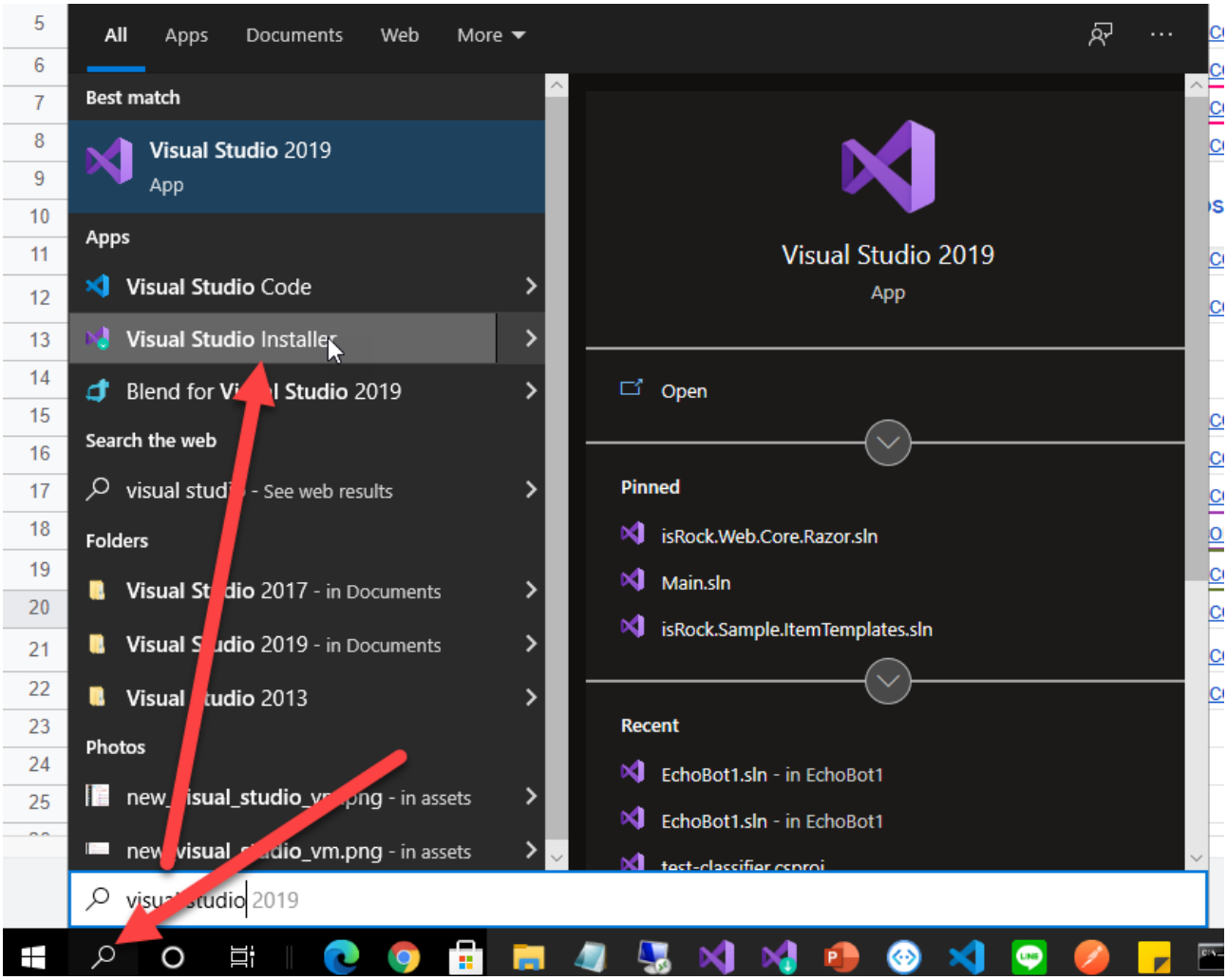
A> 下載:
[BotFramework-Emulator-4.11.0-windows-setup.exe]
(https://github.com/microsoft/BotFramework-Emulator/releases/download/v4.11.0/BotFramework-Emulator-4.11.0-windows-

setup.exe (https://github.com/microsoft/BotFramework-Emulator/releases/download/v4.11.0/BotFramework-Emulator-4.11.0-windows-setup.exe)



B> 在VS2019中安裝 Bot Framework v4 SDK Templates for Visual Studio, 位於 https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4 (https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4))

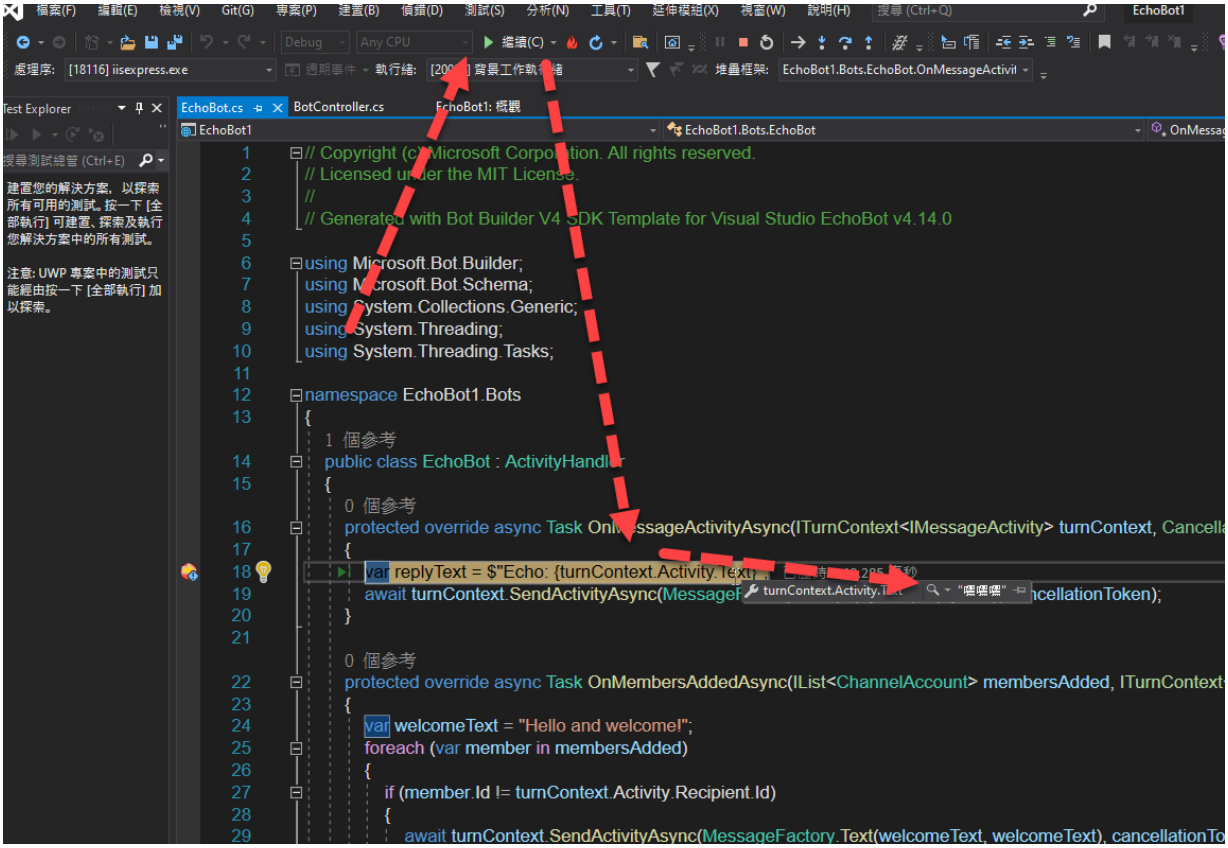
更新 VS 2019:



Lab 14: 使用emulator測試 bot

1. 建立echo bot
2. 執行
3. 開啟 emulator
4. 與bot對話

5. 在VS2019中，設定中斷點



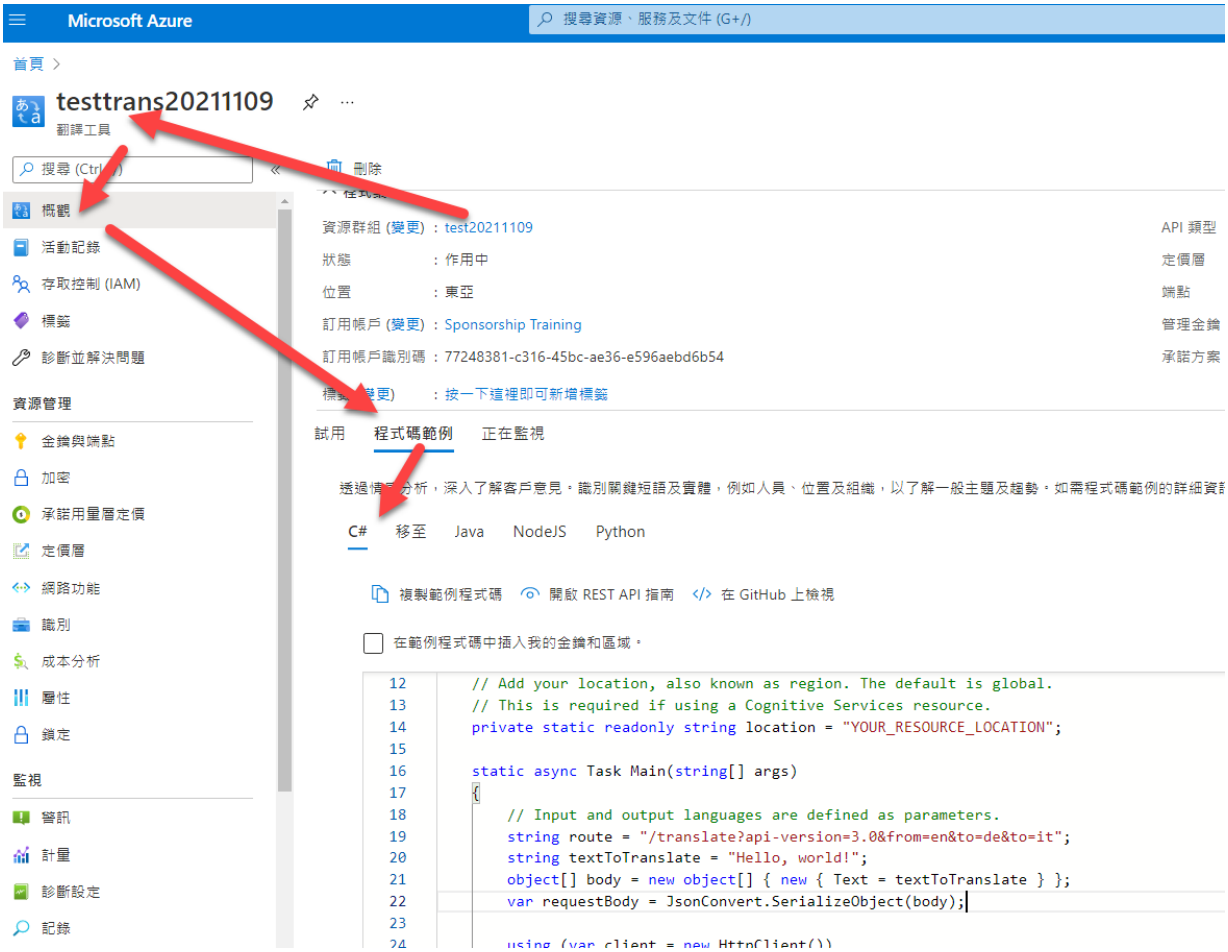
6. 觀察中斷點中，用戶傳來訊息

Lab 15:建立一個翻譯bot

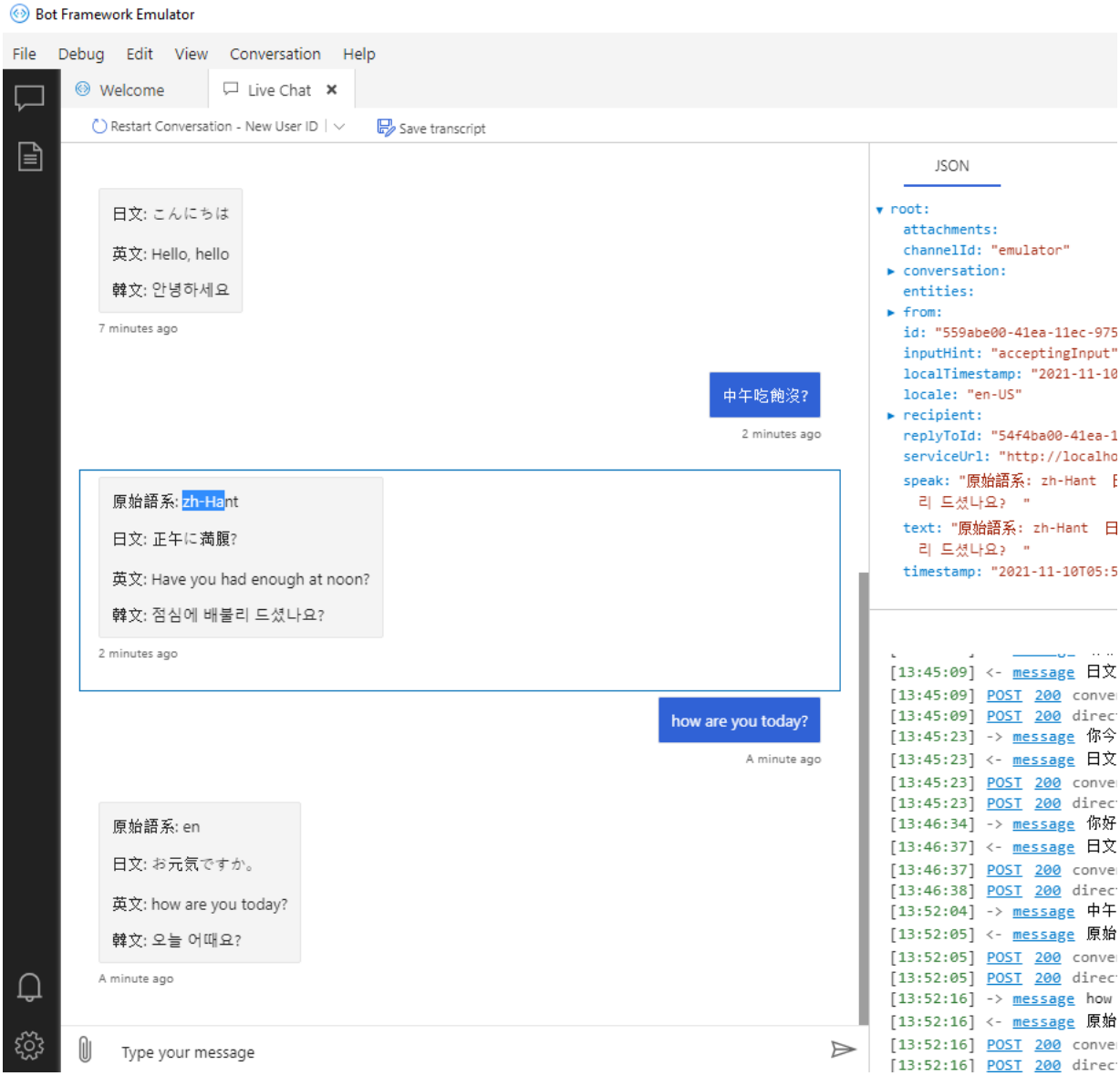
步驟:

1. 在VS2019中建立 bot 專案(如果沒有，則需安裝 Bot Framework v4 SDK Templates for Visual Studio, 位於 https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4 (https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4))
2. 啟動bot
3. 開啟Emulator
4. 設定bot connect
5. 在Emulator測試環境中與bot對話...

6. 修改程式碼，當用戶輸入任何文字時，Bot將其翻譯為日、英、韓文...



截圖:



sample code:

```
public class EchoBot : ActivityHandler
  {
      protected override async Task OnMessageActivityAsync(ITurnContext<
      {
          var response = MakeTranslator(turnContext.Activity.Text);
          var replyText = $"原始語系: {   response[0].detectedLanguage.la
          replyText += $"日文: {   response[0].translations[0].text} { Sy
          replyText += $"英文: {   response[0].translations[1].text} { Sy
          replyText += $"韓文: {   response[0].translations[2].text} { Sy
          await turnContext.SendActivityAsync(MessageFactory.Text(replyT
      }

      static List<TranslatorResponse> MakeTranslator(string msg)
      {
          HttpClient client = new HttpClient();
          string endpoint = "https://api.cognitive.microsofttranslator.c

          client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
          client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Region

          var JsonBody = "[{'text':'" + msg + "'}]";
          var content =
             new StringContent(JsonBody, System.Text.Encoding.UTF8, "app
          var response = client.PostAsync(endpoint, content).Result;
          var ResponseJson = response.Content.ReadAsStringAsync().Result
          return Newtonsoft.Json.JsonConvert.DeserializeObject<List<Tran

      }

      protected override async Task OnMembersAddedAsync(IList<ChannelAcc
      {
          var welcomeText = "Hello and welcome!";
          foreach (var member in membersAdded)
          {
              if (member.Id != turnContext.Activity.Recipient.Id)
              {
                  await turnContext.SendActivityAsync(MessageFactory.Tex
              }
          }
      }
  }

  // Root myDeserializedClass = JsonConvert.DeserializeObject<Root>(myJs
  public class DetectedLanguage
  {
      public string language { get; set; }
      public double score { get; set; }
  }

  public class Translation
  {
      public string text { get; set; }
      public string to { get; set; }
  }
```

```
55
56      public class TranslatorResponse
57      {
58          public DetectedLanguage detectedLanguage { get; set; }
59          public List<Translation> translations { get; set; }
60      }
61
```

Lab 16: 在bot中加入情緒分析功能

1. 參考 https://github.com/isdaviddong/AI-102-AzureCognitiveTextAnalysisExample/blob/main/Program.cs (https://github.com/isdaviddong/AI-102-AzureCognitiveTextAnalysisExample/blob/main/Program.cs)
2. 在 bot中加入情緒分析功能

sample code:

```
// Copyright (c) Microsoft Corporation. All rights reserved.
// Licensed under the MIT License.
//
// Generated with Bot Builder V4 SDK Template for Visual Studio EchoBot v4

using Azure;
using Azure.AI.TextAnalytics;
using Microsoft.Bot.Builder;
using Microsoft.Bot.Schema;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading;
using System.Threading.Tasks;

namespace EchoBot1.Bots
{
    public class EchoBot : ActivityHandler
    {
        protected override async Task OnMessageActivityAsync(ITurnContext<
        {
            var response = MakeTranslator(turnContext.Activity.Text);
            var replyText = $"原始語系: {   response[0].detectedLanguage.la
            replyText += $"日文: {   response[0].translations[0].text} { S
            replyText += $"英文: {   response[0].translations[1].text} { S
            replyText += $"韓文: {   response[0].translations[2].text} { S
            replyText += $"GetSentiment: {GetSentiment(turnContext.Activit
            await turnContext.SendActivityAsync(MessageFactory.Text(replyT
        }

        public static string GetSentiment(string text)
        {
            const string cogSvcKey = "798ff63e8c494c029dc2280dd5766347";
            const string cogSvcEndpoint = "https://testcog20211107.cogniti

            // Create client using endpoint and key
            AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
            Uri endpoint = new Uri(cogSvcEndpoint);
            var client = new TextAnalyticsClient(endpoint, credentials);

            // Call the service to get the detected language
            var result = client.AnalyzeSentiment(text, "zh");
            return (result.Value.Sentiment.ToString());
        }

        static List<TranslatorResponse> MakeTranslator(string msg)
        {
            HttpClient client = new HttpClient();
            string endpoint = "https://api.cognitive.microsofttranslator.c

            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Region

            var JsonBody = "[{'text':'" + msg + "'}]";
```

```
            var content =
               new StringContent(JsonBody, System.Text.Encoding.UTF8, "app
            var response = client.PostAsync(endpoint, content).Result;
            var ResponseJson = response.Content.ReadAsStringAsync().Result
            return Newtonsoft.Json.JsonConvert.DeserializeObject<List<Tran

        }

        protected override async Task OnMembersAddedAsync(IList<ChannelAcc
        {
            var welcomeText = "Hello and welcome!";
            foreach (var member in membersAdded)
            {
                if (member.Id != turnContext.Activity.Recipient.Id)
                {
                    await turnContext.SendActivityAsync(MessageFactory.Tex
                }
            }
        }
    }

    // Root myDeserializedClass = JsonConvert.DeserializeObject<Root>(myJs
    public class DetectedLanguage
    {
        public string language { get; set; }
        public double score { get; set; }
    }

    public class Translation
    {
        public string text { get; set; }
        public string to { get; set; }
    }

    public class TranslatorResponse
    {
        public DetectedLanguage detectedLanguage { get; set; }
        public List<Translation> translations { get; set; }
    }


}

```

Lab 17: Analyze Images with Computer Vision

ref: 課本lab
https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/15-computer-vision.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/15-computer-vision.html)

or

Lab 17: Implement Computer Vision with MS Bot

步驟:

1. 建立可辨識圖片的computer vision服務
2. 建立bot framework機器人(echo bot 3.1)
3. 在local以emulator運行
4. 修改程式碼加入整合圖像辨識的功能
5. 用戶可以上傳圖片，找出
    1. 圖片中有幾個人，性別，年紀
    2. 圖片的說明
    3. 圖片的標記

Lab 18: 使用 video indexer分析影片

請自行下載mp4影片，送上 video indexer (https://www.videoindexer.ai/)進行分析
https://www.videoindexer.ai/ (https://www.videoindexer.ai/)

ref: 課本lab
https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/16-video-indexer.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/16-video-indexer.html)

Lab 19: 使用Custom Vision

步驟:

1. 進入 https://www.customvision.ai/ (https://www.customvision.ai/) Portal
2. 建立Custom Vision服務
3. 從[概觀]進入Custom Vision Portal
4. 下載圖片集，使用其中的 貓 & 狗圖檔進行訓練
5. 在portal中測試
6. 使用 postman 呼叫

sample images:
https://arock.blob.core.windows.net/blogdata202109/圖片集.zip
(https://arock.blob.core.windows.net/blogdata202109/%E5%9C%96%E7%89%87%E9%9B%86.zip)

課本Lab – Classify Images with Custom Vision

https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/17-image-classification.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/17-image-classification.html)

Lab 20 : 使用Custom Vision SDK

步驟:

1. 建立dotnet core console app

```
md folder
cd folder
dotnet new console
dotnet add package Microsoft.Extensions.Configuration.Json
dotnet add package Microsoft.Azure.CognitiveServices.Vision.CustomVision.P
code .
```

or 直接clone

```
git clone https://github.com/isdaviddong/AI-102-CustomVision-classifyImage
```

3. 參考github程式
   https://github.com/isdaviddong/AI-102-CustomVision-classifyImage
   (https://github.com/isdaviddong/AI-102-CustomVision-classifyImage)
4. 建立或修改 appsettings.json
   注意

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>
  <!--注意這一塊-->
 <ItemGroup>
    <None Update="appsettings.json">
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </None>
  </ItemGroup>
  <!--注意-->
  <ItemGroup>
    <PackageReference Include="Microsoft.Azure.CognitiveServices.Vision.Cu
    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Ve
  </ItemGroup>

</Project>
```

5. 在test-images資料夾中放置圖片
6. 測試執行

7. 檢視辨識結果 <-- 截圖

```
PS D:\folder> dotnet run
test-images\22-212629-5e7dee09-c7ef-4f9a-84cd-cc17fc74f586.png: Dog (99.6%)
test-images\22-212633-ce5e5186-8712-4e41-99bd-69488f1af54d.png: Dog (99.6%)
test-images\22-212647-34c17bed-b8cb-484c-99c1-76f35cd0966b.png: Dog (98.8%)
PS D:\folder>
```

Lab 21: object detect

1. 建立 object detection專案
2. 上傳照片並標示tag
3. 訓練
4. 使用圖片偵測

課本Lab : Detect Objects in Images with Custom Vision

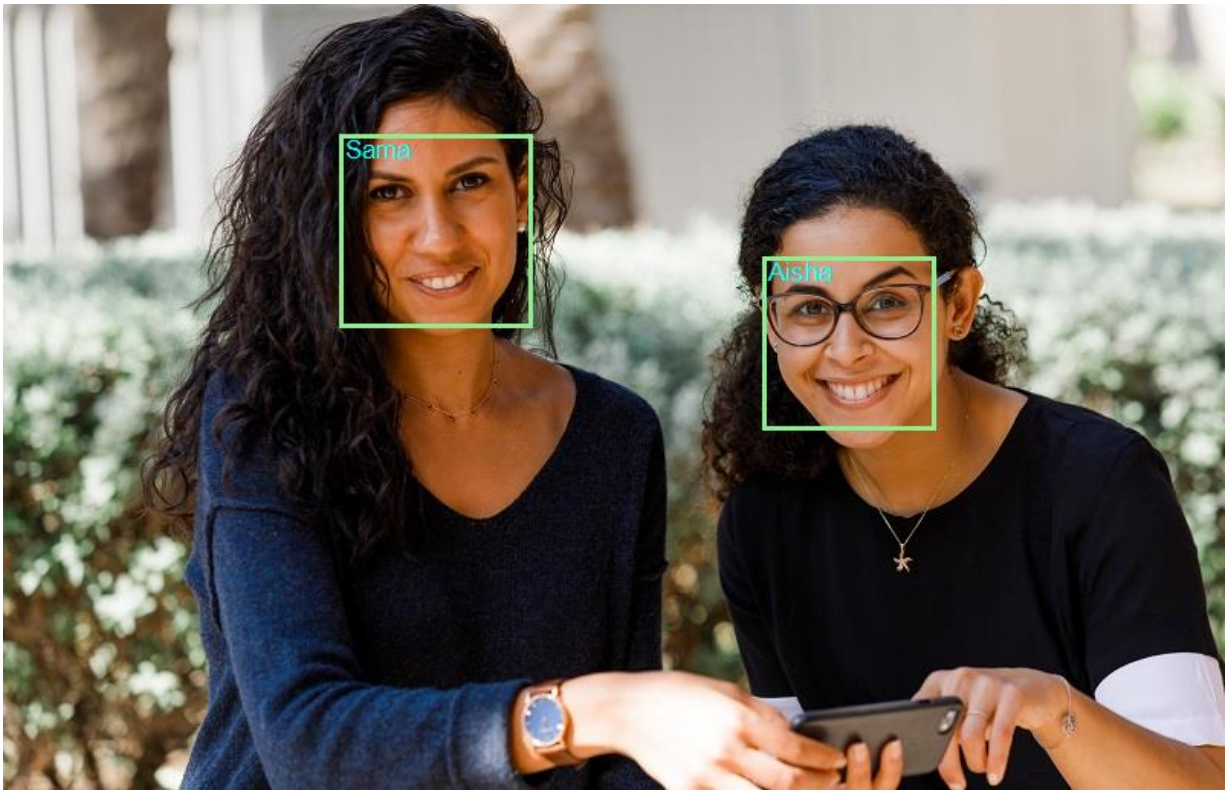
https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/18-object-detection.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/18-object-detection.html)

Lab 22:

課本Lab : https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/19-face-service.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/19-face-service.html)

1. clone 課本上的專案
2. 依照指示建立程式碼
3. 執行功能 1,2,3,4,5

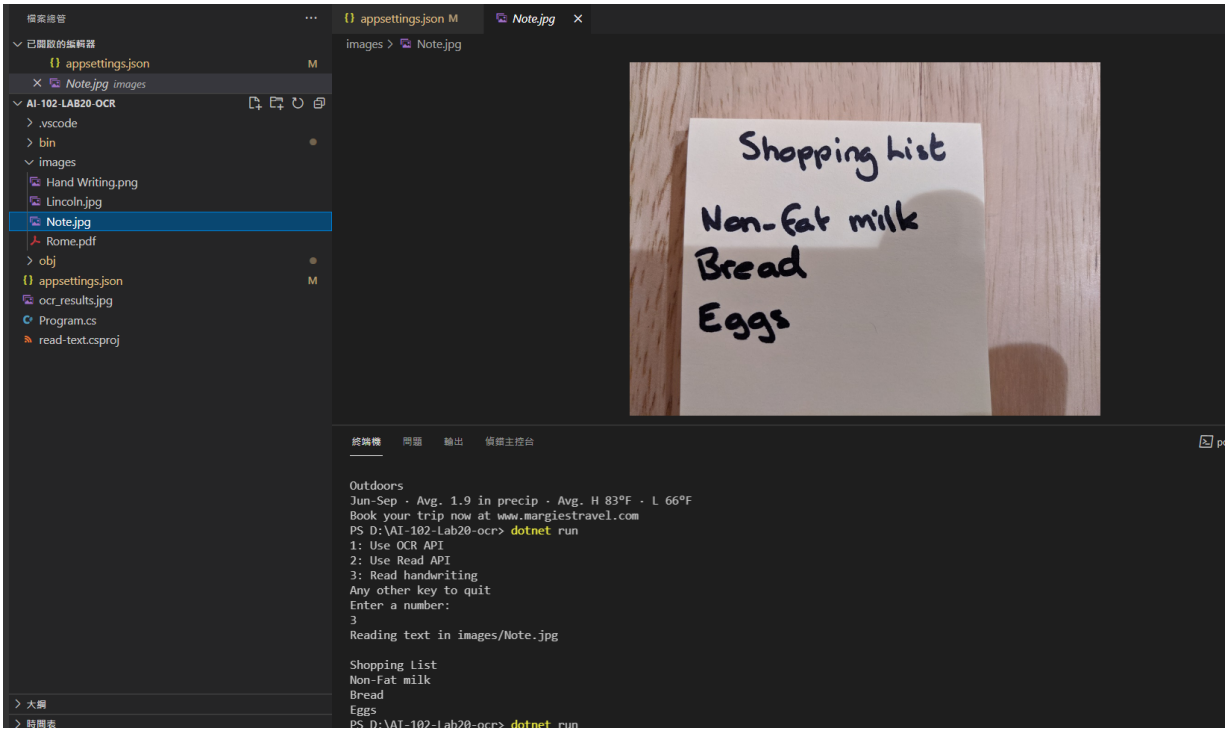
4. 檢視結果



Lab 23:

課本Lab : https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/20-ocr.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/20-ocr.html)

1. clone 課本上的專案
2. 依照指示建立程式碼
3. 執行功能 1,2,3
4. 檢視結果

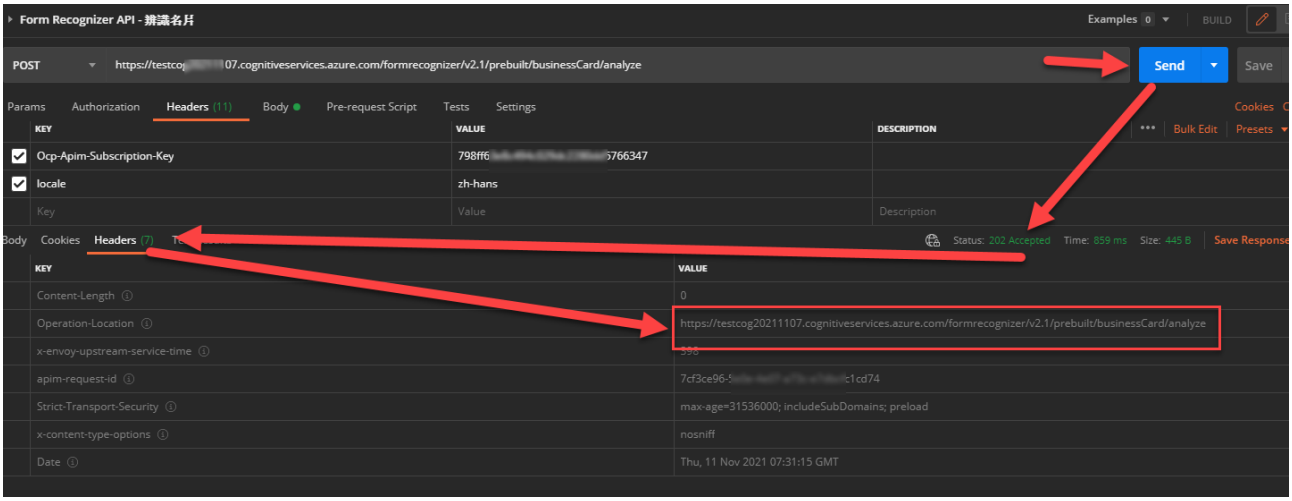


Lab 24：使用Form Recognizer API辨識名片

1. 進入 Azure Portal
2. 建立 cognitive service
3. 參考文件
   https://westus.dev.cognitive.microsoft.com/docs/services/form-recognizer-api-v2-1/operations/AnalyzeBusinessCardAsync
   (https://westus.dev.cognitive.microsoft.com/docs/services/form-recognizer-api-v2-1/operations/AnalyzeBusinessCardAsync)
4. 使用 postman 呼叫 (post)

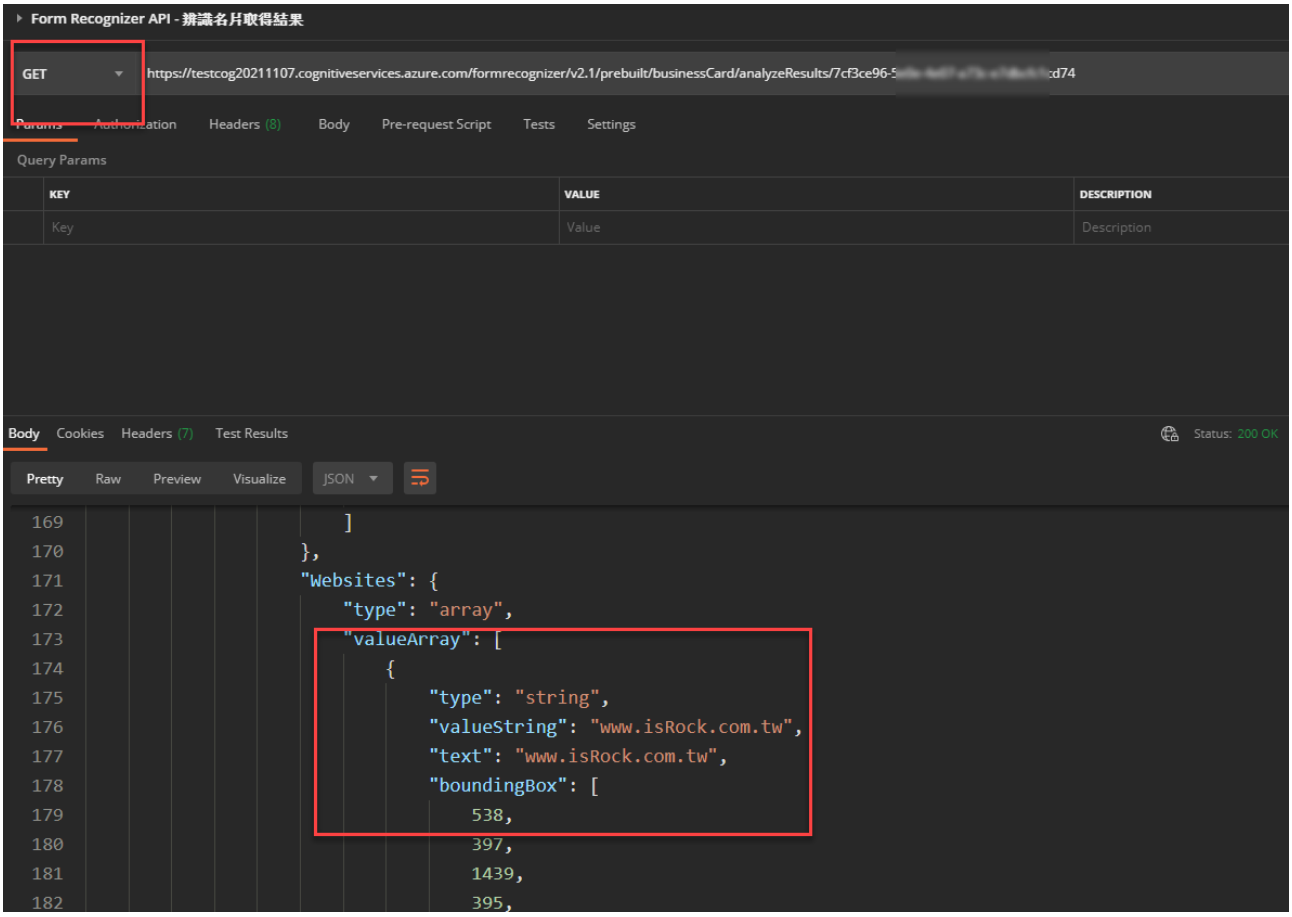
```
POST /formrecognizer/v2.1/prebuilt/businessCard/analyze HTTP/1.1
Host: testcog20211107.cognitiveservices.azure.com
Ocp-Apim-Subscription-Key: 798ff63XXXXXc029dc2280dd5766347
locale: zh-hans
Content-Type: image/jpeg
Content-Length: 22

"<file contents here>“
```



1. 辨識名片圖檔
2. 取得辨識結果 (get)

```
GET /formrecognizer/v2.1/prebuilt/businessCard/analyzeResults/46a296de-XXX
Host: testcog20211107.cognitiveservices.azure.com
Ocp-Apim-Subscription-Key: 798ff63XXXXX4c029dc2280dd5766347
```



Lab 25: Extract Data from Forms

課本Lab : https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/21-form-recognizer.html (https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/21-form-recognizer.html)

AZ CLI相關資訊:

AZ CLI : https://docs.microsoft.com/en-us/cli/azure/what-is-azure-cli (https://docs.microsoft.com/en-us/cli/azure/what-is-azure-cli)
download : https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli (https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli)

問卷

恆逸問卷:

恆逸問卷調查 https://survey.uuu.com.tw/ (https://survey.uuu.com.tw/)

code : 211025

原廠問卷:

恆逸教育訓練中心滿意度調查

親愛的顧客您好,

感謝您參加恆逸訓練課程，希望這幾天的課程能夠帶給您滿滿的收穫，請協助撥冗填寫課後滿意度問卷，
歡迎您提供珍貴的寶貴意見，您的意見會是我們進步的最大動力！

恆逸教育訓練中心 敬上

請輸入班別： 211025 開始查詢

原廠課程線上評量系統

- 微軟課程請進入 www.metricsthatmatter.com/ucominfo97 網頁
- CISCO課程請進入www.metricsthatmatter.com/ucom 網頁

由於本課程為原廠官方課程，因此除了恆逸資訊本身的課後問卷之外，另要麻煩同學使用線上評量系統填寫之官方

metrics that matter

Class List

Class list for Systex Corporation
Please select your class:

Select one

M20744 Nov 05, 2021-- Su
M20768 Nov 05, 2021-- YOUNG
M29742 Oct 31, 2021-- Tu
M29764 Nov 01, 2021-- HUANG
MAI-102T00 Nov 11, 2021-- TUNG
MAZ-204T00 Oct 29, 2021-- TSAO
MAZ-400T00 Oct 29, 2021-- TUNG

tags: AI-102 Azure