

# 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/HkSInfoUt> (<https://hackmd.io/@twdeveloper/HkSInfoUt>).

帳號建立:

MS Account:

<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

# Exercise



## check list

- ```

1 dotnet --version
2 git --version
3 docker --version
  
```

```
PS C:\Users\vmadmin> dotnet --version
5.0.402
PS C:\Users\vmadmin> git --version
git version 2.33.0.windows.2
PS C:\Users\vmadmin> docker --version
Docker version 20.10.8, build 3967b7d
PS C:\Users\vmadmin> |
```

- VS 2019
- VS Code
- docker desktop
- dotnet 3.1+
- git
- 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  
<https://westcentralus.dev.cognitive.microsoft.com/docs/services/computer-vision-v3-1-ga/operations/5d986960601faab4bf452005>
3. 查找文件
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/rU4l3nRGaxw> (<https://youtu.be/rU4l3nRGaxw>)

## 建立資源:

首頁 > 建立資源 > 電腦視覺 >

# Create Computer Vision

**Basics** Network Identity Tags 檢閱 + 建立

Boost content discoverability, accelerate text extraction, and create products that more people can use by embedding vision capabilities in your apps. Use visual data processing to label content (from objects to concepts), extract printed and handwritten text, recognize familiar subjects like brands and landmarks, and moderate content. No machine learning expertise is required. [Learn more](#).

**Project Details**

|        |                                                        |
|--------|--------------------------------------------------------|
| 訂用帳戶 * | Azure Pass - 贊助 (5614f07e-5460-4cb1-802d-b7bdb0252c90) |
| 資源群組 * | test20211108                                           |
|        | <a href="#">新建</a>                                     |

## Instance Details

|                                                                                                                                                                                                                                             |              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| 區域                                                                                                                                                                                                                                          | 東南亞          |
| Name *                                                                                                                                                                                                                                      | test20211108 |
| <span style="color: red;">✖</span> The sub-domain name is already used. Please pick a different name.<br><span style="color: red;">✖</span> The provided sub-domain name is either invalid or already in use. Please pick a different name. |              |

可以用的圖片(JSON):

```
1 | {"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.

```

1  {
2      "CognitiveServicesEndpoint": "YOUR_COGNITIVE_SERVICES_ENDPOINT",
3      "CognitiveServiceKey": "YOUR_COGNITIVE_SERVICES_KEY"
4  }

```

## 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:

```

1 #建立資料夾
2 md 資料夾名稱
3 #git版控指 . 複製github雲端檔案
4 git clone https://github.com/MicrosoftLearning/AI-102-AIEngineer
5 #進入資料夾
6 cd 資料夾名稱
7 #開啟VS code
8 Code .
9 #安裝nuget套件
10 dotnet add package Azure.AI.TextAnalytics
11 #build程式
12 dotnet build
13 #執行程式
14 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

```
1 docker run --rm -it -p 127.0.0.1:5000:5000 --memory 4g --cpus 1 mcr.microsoft.com/azure-ai-textanalytics
```

# 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
```

```
1  {
2      "documents": [
3          {
4
5              "id": "1",
6              "text": "你他媽的這什麼鬼？"
7          },
8          {
9
10             "id": "2",
11             "text": "早安唷，你老闆在嗎？"
12         },
13         {
14
15             "id": "3",
16             "text": "叫你老闆給我滾出來"
17         },
18         {
19
20             "id": "4",
21             "text": "La carretera estaba atascada. Había mucho tráfico el
22         }
23     ]
24 }
```



The screenshot shows a POST request in Postman to the endpoint `http://127.0.0.1:5000/text/analytics/v3.0/languages`. The request body is a JSON array containing two objects. The second object has an `id` of "4" and a `text` of "La carretera estaba atascada. Había mucho tráfico el dia de ayer.". The response is a JSON object with an `id` of "4", a `detectedLanguage` of "es", a `name` of "Spanish", an `iso6391Name` of "es", and a `confidenceScore` of 1.0.

```
POST http://127.0.0.1:5000/text/analytics/v3.0/languages
```

```
[{"id": "3", "text": "叫你老闆給我滾出來"}, {"id": "4", "text": "La carretera estaba atascada. Había mucho tráfico el dia de ayer."}]
```

```
{ "id": "4", "detectedLanguage": "es", "name": "Spanish", "iso6391Name": "es", "confidenceScore": 1.0}, {"warnings": []}
```

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>).

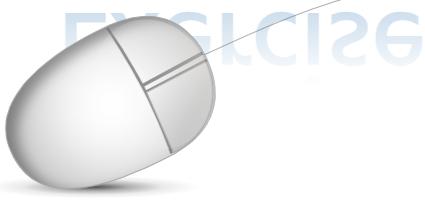
## Lab 4:

目標:

分析文字情緒

步驟:

# Exercise



1. 建立 Text Analytics 服務
2. 取得 endpoint & Key
3. 查看文件

<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:

POST https://testcog20211107.cognitiveservices.azure.com/text/analytics/v3.0/sentiment

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1  {
2      "documents": [
3          {
4              "language": "zh",
5              "id": "1",
6              "text": "乾脆去死一死好了"
7          ],
8          {
9              "language": "zh",
10             "id": "2",
11             "text": "早安唷，你老闆在嗎？"
12         }
13     ]
14 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

3  {
4      "id": "1",
5      "sentiment": "negative",
6      "confidenceScores": {
7          "positive": 0.05,
8          "neutral": 0.0,
9          "negative": 0.95
10     },
11     "sentences": [
12         {
13             "text": "乾脆去死一死好了",
14             "sentiment": "negative",
15             "confidenceScores": {
16                 "positive": 0.05,
17                 "neutral": 0.0,
18                 "negative": 0.95
19             }
20         }
21     ]
22 }
```

目標:

擷取關鍵字詞

步驟:

1. 建立 Text Analytics 服務

2. 取得 endpoint &amp; Key

3. 查看文件

<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

# Exercise



result:

Postman screenshot showing the JSON response from the Text Analytics API. A red arrow points from the 'id' field in the first document to the 'keyPhrases' field in the second document.

```

1 {
2   "documents": [
3     {
4       "language": "zh",
5       "id": "1",
6       "text": "我需要一張從台北到高雄的車票"
7     },
8     {
9       "language": "zh",
10      "id": "2",
11      ...
12    }
13  ]
14}
  
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↗

```

1 {
2   "documents": [
3     {
4       "id": "1",
5       "keyPhrases": [
6         "台北",
7         "高雄",
8         "車票"
9       ],
10      "warnings": []
11    }
12  ]
13}
  
```

參考JSON:

```
1  {
2      "documents": [
3          {
4              "id": "1",
5              "text": "我需要一張從臺北到高雄的車票"
6          },
7          {
8              "id": "2",
9              "text": "早安唷，你老闆在嗎？"
10         },
11         {
12             "id": "3",
13             "text": "叫你老闆給我滾出來"
14         },
15         {
16             "id": "4",
17             "text": "La carretera estaba atascada. Había mucho tráfico el"
18         }
19     ]
20 }
```

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 撷取關  
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)

# Exercise



```
1 #建立new console app
2 dotnet new console
3 #開啟 VS Code
4 code .
5 #裝套件
6 dotnet add package Azure.AI.TextAnalytics
```

sample code:

```
1  using System;
2  using Azure;
3  using Microsoft.Extensions.Configuration;
4  using System.Text;
5  using Azure.AI.TextAnalytics;
6
7  namespace textananalysis
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.InputEncoding = Encoding.Unicode;
14             Console.OutputEncoding = Encoding.Unicode;
15
16             Console.WriteLine("請輸入句子:");
17             var text = Console.ReadLine();
18             var Language = TextAnalytics.GetLanguage(text);
19             Console.WriteLine("Get Language:" + Language);
20             var ret = TextAnalytics.GetSentiment(text);
21             Console.WriteLine("Get Sentiment:" + ret);
22             var KeyPhrases = TextAnalytics.ExtractKeyPhrases(text);
23             foreach (var item in KeyPhrases.Value)
24             {
25                 Console.WriteLine("KeyPhrases:" + item);
26             }
27         }
28     }
29
30     public class TextAnalytics
31     {
32         const string cogSvcKey = "439c1c08a372445cb9b417c6d3d291d3";
33         const string cogSvcEndpoint = "https://mycog20211108.cognitiveserv
34
35         public static string GetLanguage(string text)
36         {
37
38             // Create client using endpoint and key
39             AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
40             Uri endpoint = new Uri(cogSvcEndpoint);
41             var client = new TextAnalyticsClient(endpoint, credentials);
42
43             // Call the service to get the detected language
44             DetectedLanguage detectedLanguage = client.DetectLanguage(text
45             return (detectedLanguage.Name);
46
47         }
48
49         public static string GetSentiment(string text)
50         {
51             // Create client using endpoint and key
52             AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
53             Uri endpoint = new Uri(cogSvcEndpoint);
54             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"

# Exercise

# EXERCISE



sample code(http)

```
1 POST /translate?api-version=3.0&to=ja&to=en&to=ko HTTP/1.1
2 Host: api.cognitive.microsofttranslator.com
3 Ocp-Apim-Subscription-Key: 428b68baaa044d1a9ef195a52a374326
4 Ocp-Apim-Subscription-Region: eastasia
5 Content-Type: application/json
6 Content-Length: 92
7
8 [
9   {"Text": "嗨，大家今天過的好嗎?"},
10  {"Text": "端午節快到了，我想吃肉粽"},
11  {"Text": "雙十國慶快到了，我想要出國玩"}
12 ]
```

## 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/>).

Speech Studio

### 語音轉換文字

快速且準確地將音訊轉譯成超過 100 種語言和變體的文字。 使用自訂語音建立的自訂模型，增強網域特定術語的準確性，並克服背景雜音、口音或獨特詞彙等語音  
[深入了解語音轉換文字](#)



### 文字轉換語音

從 250 種語言及 70 種語言和變體文字中選擇，組建可自然說話的應用程式與服務。使用自訂的語音來區別您的品牌，並使用不同的說話樣式和情緒語調來存取語音  
[深入了解文字轉換語音](#)



## 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>  
[\(<https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/07-speech.html>\).](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

```

1 [
2   "query": "我要店員的電話",
3   "prediction": {
4     "topIntent": "客訴",
5     "intents": {
6       "客訴": {
7         "score": 0.9155762
8       },
9       "None": {
10        "score": 0.032084506
11      },
12      "點餐行為": {
13        "score": 0.031059565
14      }
15    }
16  }
17]
  
```

## 關於Entity Type

使用LUIS

*entity(地點)*

- 我要去**高雄**旅遊
- 買一張從**台北**到**左營**的車票

*Entity(Hierarchical)*  
地點:出發地

*Entity(Hierarchical)*  
地點:目的地

## 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>).

```
1 #create folder
2 md testluisapp
3 #into fondler
4 cd testluisapp
5 #create console app
6 dotnet new console
7 #add nuget pacakge
8 dotnet add package Microsoft.Azure.CognitiveServices.Language.LUIS.Runtime
9 #run this app
10 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)

[首頁](#) > [新增](#) > [QnA Maker](#) >

### 建立

認知服務

\* 基本 標籤 檢閱 + 建立

#### 專案詳細資料

|      |               |
|------|---------------|
| 訂用帳戶 | AzurePass2021 |
| 資源群組 | testqna2021   |
| 名稱   | testqna2021   |
| 定價層  | S0            |

#### Azure 搜尋服務詳細資料 - 用於資料

|               |    |
|---------------|----|
| Azure 搜尋服務位置  | 東亞 |
| Azure 搜尋服務定價層 | S  |

#### App Service 詳細資料 - 用於執行階段

|        |             |
|--------|-------------|
| 應用程式名稱 | testqna2021 |
| 網站位置   | 東亞          |

#### 應用程式見解詳細資料 - 用於監測及聊天記錄

|        |     |
|--------|-----|
| 應用程式見解 | 已停用 |
|--------|-----|

正在驗證...

< 上一步：標籤

[下載自動化的範本](#)

## 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)  
<sub>([https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35\\_bDBeHfA](https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA))</sub>

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

Use the below HTTP request to call your Knowledgebase. [Learn more.](#)

Postman    Curl

```
POST /knowledgebase/generateAnswer
Host: https://testqna2021.azurewebsites.net/qnamaker
Authorization: EndpointKey 875090dc-XXXX-XXXX-a2bf-0027e39fce53
Content-Type: application/json
{"question":<Your question>"}
```

## B> 在postman中呼叫

Untitled Request

POST https://testqna2021.azurewebsites.net/qnamaker/generateAnswer ...

Params Authorization Headers (11) Body (1) Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {"question": "卡片找不到了"}

Body Cookies (2) Headers (13) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
  "answers": [
    {
      "questions": [
        "好市多會員卡遺失了如何申請補發?",
        "卡片遺失了",
        "卡片掉了...",
        "卡片不見了"
      ],
      "answer": "請您攜帶有照片之身分證明文件，親至鄰近的賣場會員部櫃台辦理補發。不需負擔任何費用，我們立即換發新卡給您（原舊卡於補發新卡時立即失
```

Status: 200 OK Time: 219 ms Size: 905 B Save Response

```
1 curl --location --request POST 'https://tXXXXXXXXX21.azurewebsites.net/qnamaker/generateAnswer'
2 --header 'Authorization: EndpointKey 875090dc-XXXX-XXXX-a2bf-0027e39fce53'
3 --header 'Content-Type: application/json' \
4 --data-binary '{"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:

- 開啟Azure Portal，建立QnA Maker服務

<https://portal.azure.com/#create/Microsoft.CognitiveServicesQnAMaker>  
[\(https://portal.azure.com/#create/Microsoft.CognitiveServicesQnAMaker\)](https://portal.azure.com/#create/Microsoft.CognitiveServicesQnAMaker)

- QnA Maker Console (<http://qnamaker.ai> (<http://qnamaker.ai>))

- 建立KM

<https://www.qnamaker.ai/Create> (<https://www.qnamaker.ai/Create>)

- 匯入QA

- 發佈成為服務

[https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35\\_bDBeHfA](https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA)  
[\(https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35\\_bDBeHfA\)](https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA)

- 使用postman呼叫 <-截圖

```

1 curl --location --request POST 'https://tXXXXXXXXX21.azurewebsites.net/c
2   --header 'Authorization: EndpointKey 875090dc-XXXX-XXXX-a2bf-0027e39fc
3   --header 'Content-Type: application/json' \
4   --data-raw '{"question":"打疫苗是否危險?"}'

```

The screenshot shows a Postman request to <https://testqna20211109.azurewebsites.net/qnamaker/knowledgebases/d32903c-68b7-41d3-8772-07bad41dd788/generateAnswer>. The request method is POST, and the body is a JSON object with a single key-value pair: {"question": "什麼時候才會該我?"}. The response status is 200 OK, and the response body is a JSON object containing the question, answer, score, id, and source.

| Index | Value                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4     | "questions": [                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 5     | "6 什麼時候輪到我接種？"                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 6     | ],                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 7     | "answer": "在COVID-19疫苗接種計畫實施初期，疫苗的供應量受到限制，故需擬定優先接種對象並依序施行。考量目前國內COVID-19疫情屬低社區風險，為維持醫療量能、防疫量能、維持社會運作、維護長者及高風險群健康，中央流行疫情指揮中心依衛生福利部傳染病防治諮詢會預防接種組建議，擬定10大類對象為接種對象。對象及優先順序為：\n\n1. 醫事人員\n\n2. 中央及地方政府防疫人員\n\n3. 高接觸風險第一線工作人員\n\n4. 因特殊情形必要出國者\n\n5. 維持機構及社福照顧系統運作\n\n6. 75歲以上長者(具原住民身分者為65歲以上)\n\n7. 維持國家社會機能正常運作者\n\n8. 65歲以上長者\n\n9. 可能增加感染及疾病嚴重風險之疾病風險族群\n\n10. 50-64歲成人指揮中心將視疫情及狀況，滾動檢討COVID-19疫苗接種對象。 * 2018-07-29 最後更新日期 2021/6/12" , |
| 8     | "score": 42.94,                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 9     | "id": 6,                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 10    | "source": "https://www.cdc.gov.tw/Category/QAPage/JCyOJznV52tt35_bDBeHfA"                                                                                                                                                                                                                                                                                                                                                               |

## 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 step hand-on lab:

<https://github.com/isdaviddong/HOL-LineBotSDK/blob/master/webhook/Lab%2011%20-%20如何建立可Echo的LINE%20Bot.md> (<https://github.com/isdaviddong/HOL-LineBotSDK/blob/master/webhook/Lab%2011%20-%20如何建立可Echo的LINE%20Bot.md>)

```

1 # script to create .net core web api project
2 md folder
3 cd folder
4 dotnet new webapi
5 dotnet add package linebotsdk
6 dotnet new --install isRock.Template.LineWebHook::1.0.12
7 code .
8 dotnet new linewebhook
9 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

The screenshot shows the Microsoft QnA Maker Console interface. At the top, it says "testqna20211109-bot | 在網路聊天中測試 ...". On the left, there's a sidebar with navigation links: 概觀, 活動記錄, 存取控制 (IAM), 標籤, 設定, Bot 設定檔, 組態, 頻道, Channel (deprecated), 定價, 在網路聊天中測試 (highlighted with a red arrow), 加密, 屬性, 鎮定, 監視, 交談分析, 聲訊, and 告警. The main area shows a conversation window with a message from the bot: "Hello and welcome!" followed by "剛剛". Below the message is a list of bullet points about COVID-19 vaccines. A user message "疫苗安全嗎?" is shown on the right, with a red arrow pointing to it. The bottom of the screen has an input field "輸入您的訊息" and a send button "⇒".

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>).

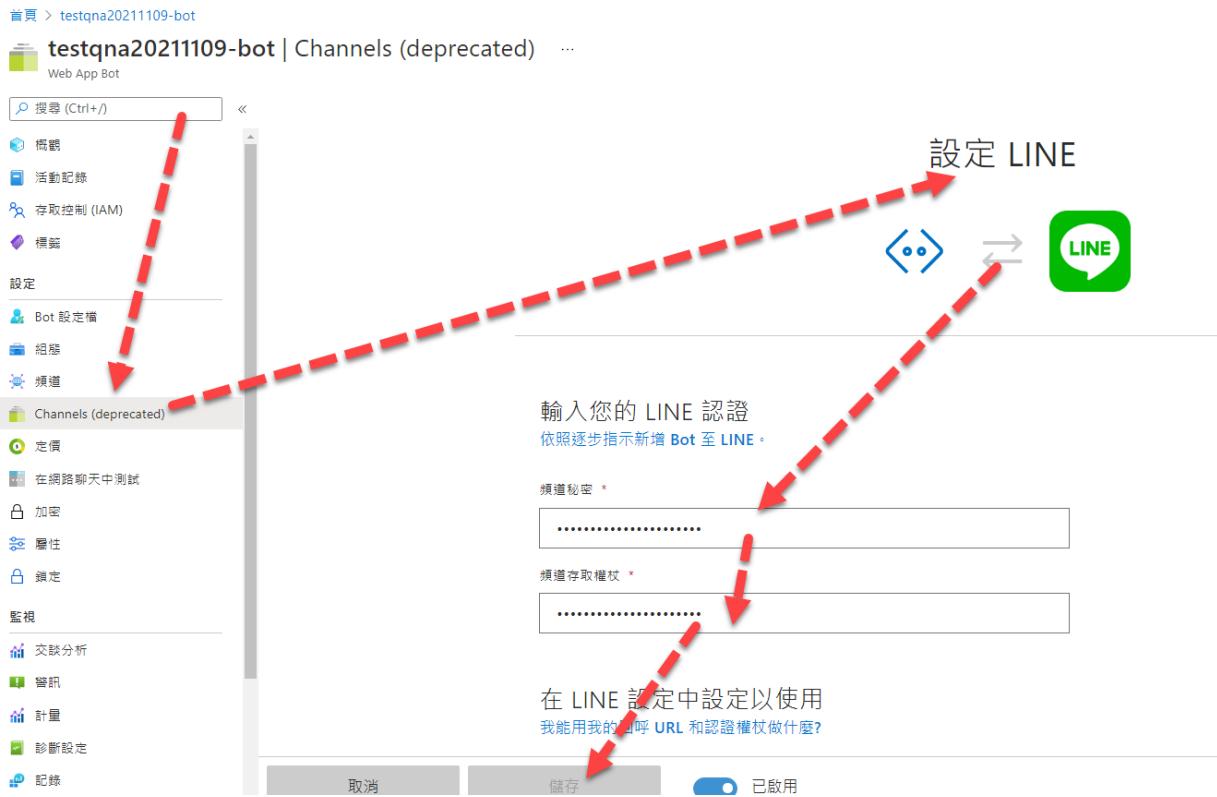
建立第一支LINE Bot (2020版)

<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

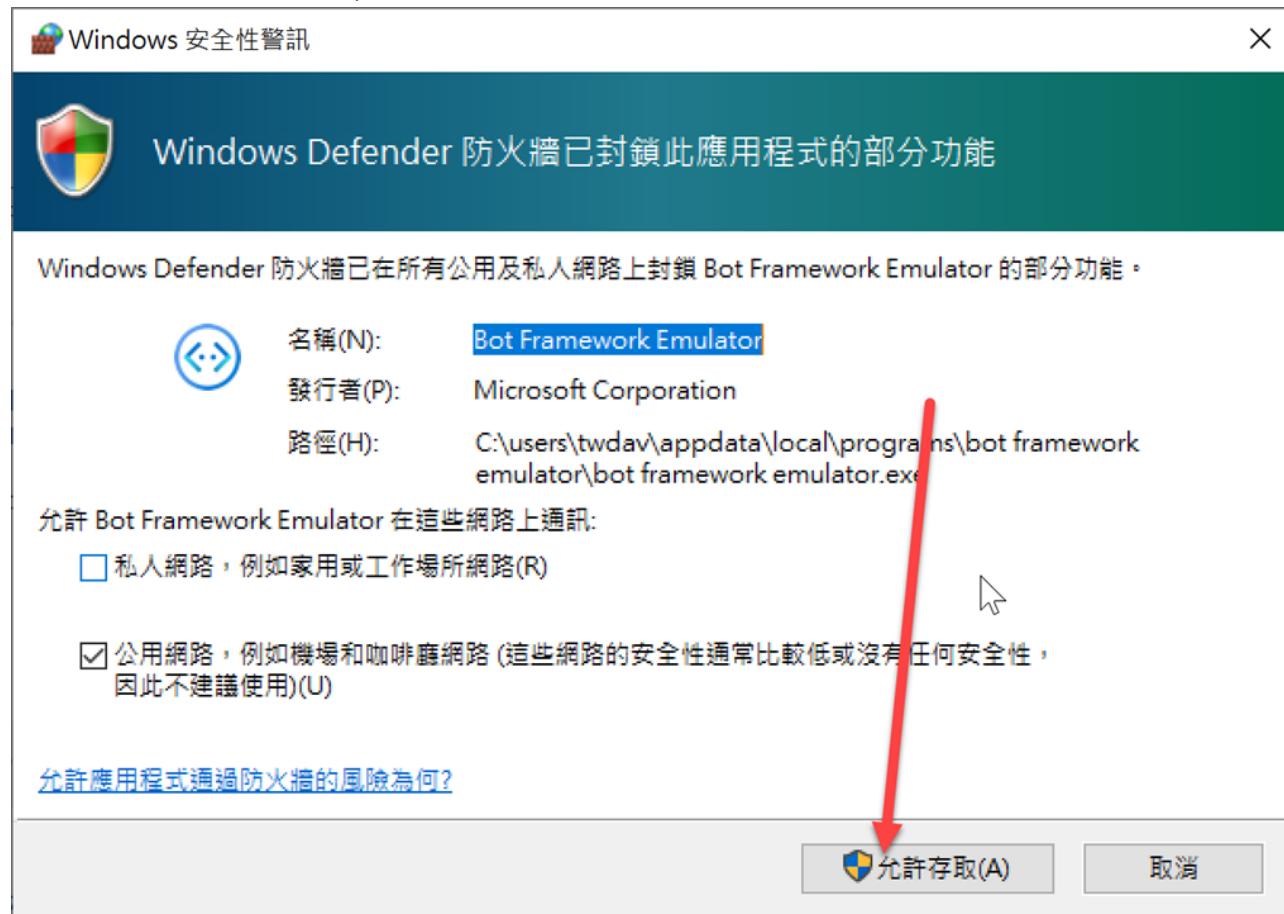
**環境準備:**

A> 下載:

[BotFramework-Emulator-4.11.0-windows-setup.exe]

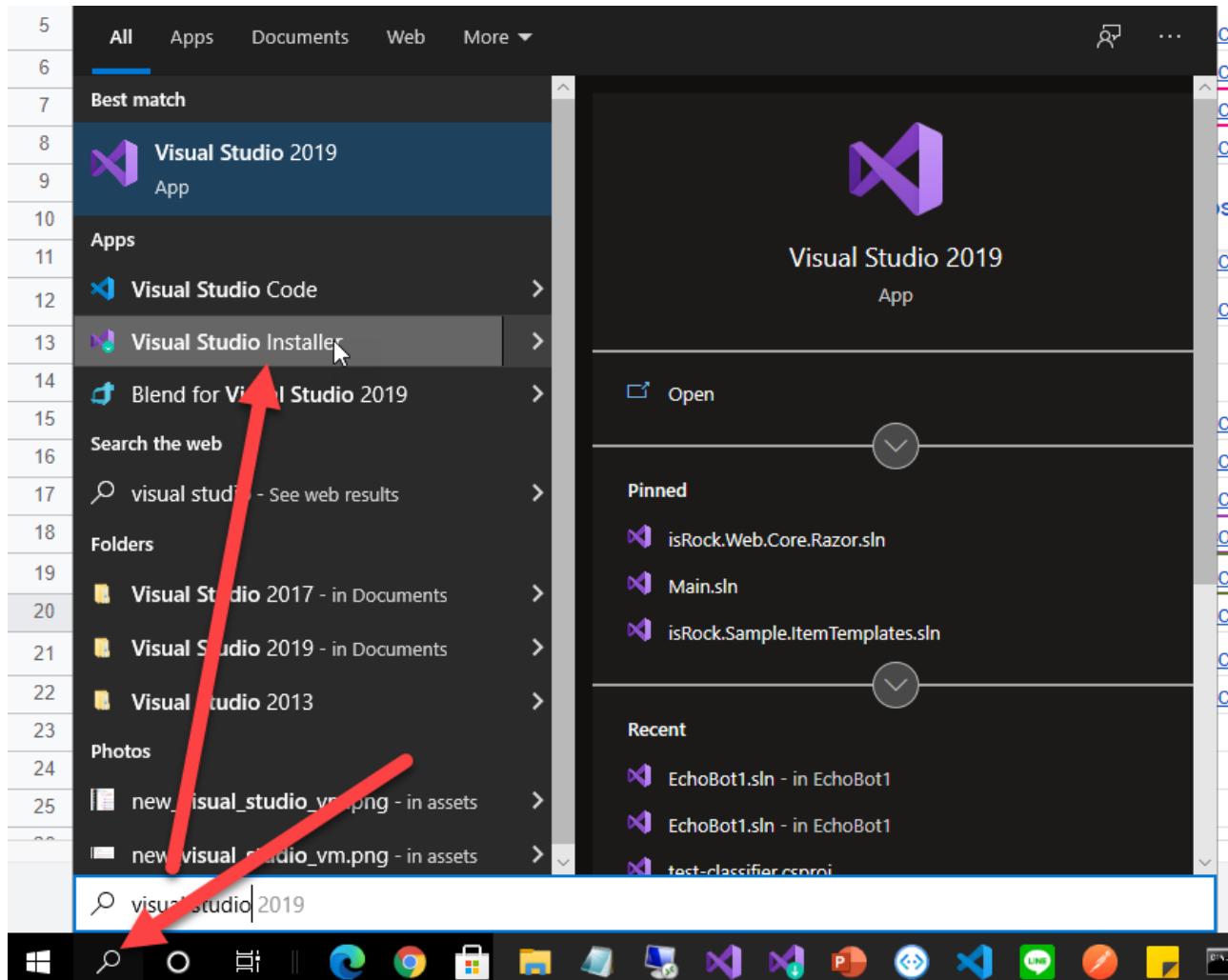
(<https://github.com/microsoft/BotFramework-Emulator/releases/download/v4.11.0/BotFramework-Emulator-4.11.0-windows->

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

```

1 // Copyright (c) Microsoft Corporation. All rights reserved.
2 // Licensed under the MIT Licence.
3 //
4 // Generated with Bot Builder V4 SDK Template for Visual Studio EchoBot v4.14.0
5
6 using Microsoft.Bot.Builder;
7 using Microsoft.Bot.Schema;
8 using System.Collections.Generic;
9 using System.Threading;
10 using System.Threading.Tasks;
11
12 namespace EchoBot1.Bots
13 {
14     1 個參考
15     public class EchoBot : ActivityHandler
16     {
17         0 個參考
18         protected override async Task OnMessageActivityAsync(ITurnContext< IMessageActivity> turnContext, CancellationToken cancellationToken)
19         {
20             var replyText = $"Echo: {turnContext.Activity.Text}";
21             await turnContext.SendActivityAsync(MessageFactory.Text(replyText), cancellationToken);
22         }
23     }
24
25     0 個參考
26     protected override async Task OnMembersAddedAsync(IList< ChannelAccount > membersAdded, ITurnContext< IMemberActivity> turnContext, CancellationToken cancellationToken)
27     {
28         var welcomeText = "Hello and welcome!";
29         foreach (var member in membersAdded)
30         {
31             if (member.Id != turnContext.Activity.Recipient.Id)
32             {
33                 await turnContext.SendActivityAsync(MessageFactory.Text(welcomeText, welcomeText), cancellationToken);
34             }
35         }
36     }
37 }
38
39 
```

## 6. 觀察中斷點中，用戶傳來訊息

### Lab 15:建立一個翻譯bot

步驟：

- 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>))
- 啟動bot
- 開啟Emulator
- 設定bot connect
- 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...

The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the Microsoft Azure logo and a search bar containing the text '搜尋 (Ctrl + F)'. Below the header, the main content area has a title 'testtrans20211109' with a red arrow pointing to it. To the right of the title, there are several tabs: '概述' (Overview), '活動記錄' (Activity Log), '存取控制 (IAM)', '標籤', and '診斷並解決問題'. Under the '資源管理' (Resource Management) section, there are links for '金鑰與端點', '加密', '承諾用量層定價', '定價層', '網路功能', '識別', '成本分析', '屬性', and '鎖定'. On the right side, there are sections for 'API 類型' (API Type), '定價層' (Pricing Tier), '端點' (Endpoint), '管理金鑰' (Manage Key), and '承諾方案' (Commitment Plan). The main content area displays the resource details: 資源群組 (變更) : test20211109, 狀態 : 作用中, 位置 : 東亞, 訂用帳戶 (變更) : Sponsorship Training, 訂用帳戶識別碼 : 77248381-c316-45bc-ae36-e596aebd6b54. Below these details, there's a note: '標籤 (變更) : 按一下這裡即可新增標籤'. Under the '試用' (Try) section, the '程式碼範例' (Code Example) tab is selected, indicated by a red arrow. It shows code examples for C#, Java, NodeJS, and Python. The C# example is displayed in a code editor:

```
// Add your location, also known as region. The default is global.  
// This is required if using a Cognitive Services resource.  
private static readonly string location = "YOUR_RESOURCE_LOCATION";  
  
static async Task Main(string[] args)  
{  
    // Input and output languages are defined as parameters.  
    string route = "/translate?api-version=3.0&from=en&to=de&to=it";  
    string textToTranslate = "Hello, world!";  
    object[] body = new object[] { new { Text = textToTranslate } };  
    var requestBody = JsonConvert.SerializeObject(body);  
  
    using (var client = new HttpClient())
```

截圖:

Bot Framework Emulator

File Debug Edit View Conversation Help

Welcome Live Chat ×

Restart Conversation - New User ID | Save transcript

JSON

```

▼ root:
  attachments:
  channelId: "emulator"
  ▶ conversation:
    entities:
    ▶ from:
      id: "559abe00-41ea-11ec-975"
      inputHint: "acceptingInput"
      localTimestamp: "2021-11-10"
      locale: "en-US"
    ▶ recipient:
      replyToId: "54f4ba00-41ea-1"
      serviceUrl: "http://localho"
      speak: "原始語系: zh-Hant ["
      text: "原始語系: zh-Hant 日"
      timestamp: "2021-11-10T05:5"

```

日文: こんにちは  
英文: Hello, hello  
韓文: 안녕하세요

7 minutes ago

中午吃饱没?

2 minutes ago

原始語系: zh-Hant  
日文: 正午に満腹?  
英文: Have you had enough at noon?  
韓文: 점심에 배불리 드셨나요?

2 minutes ago

how are you today?

A minute ago

原始語系: en  
日文: お元気ですか。  
英文: how are you today?  
韓文: 오늘 어때요?

A minute ago

Type your message ➤

```

[13:45:09] <- message 日文
[13:45:09] POST 200 conver
[13:45:09] POST 200 direc
[13:45:23] -> message 你今
[13:45:23] <- message 日文
[13:45:23] POST 200 conver
[13:45:23] POST 200 direc
[13:46:34] -> message 你好
[13:46:37] <- message 日文
[13:46:37] POST 200 conver
[13:46:38] POST 200 direc
[13:52:04] -> message 中午
[13:52:05] <- message 原始
[13:52:05] POST 200 conver
[13:52:05] POST 200 direc
[13:52:16] -> message how
[13:52:16] <- message 原始
[13:52:16] POST 200 conver
[13:52:16] POST 200 direc

```

sample code:

```
1  public class EchoBot : ActivityHandler
2  {
3      protected override async Task OnMessageActivityAsync(ITurnContext<
4      {
5          var response = MakeTranslator(turnContext.Activity.Text);
6          var replyText = $"原始語系: { response[0].detectedLanguage.la
7          replyText += $"日文: { response[0].translations[0].text } { $)
8          replyText += $"英文: { response[0].translations[1].text } { $)
9          replyText += $"韓文: { response[0].translations[2].text } { $)
10         await turnContext.SendActivityAsync(MessageFactory.Text(replyT
11     }
12
13     static List<TranslatorResponse> MakeTranslator(string msg)
14     {
15         HttpClient client = new HttpClient();
16         string endpoint = "https://api.cognitive.microsofttranslator.c
17
18         client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
19         client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Region
20
21         var JsonBody = "[{'text':'" + msg + "'}]";
22         var content =
23             new StringContent(JsonBody, System.Text.Encoding.UTF8, "app
24         var response = client.PostAsync(endpoint, content).Result;
25         var ResponseJson = response.Content.ReadAsStringAsync().Result
26         return Newtonsoft.Json.JsonConvert.DeserializeObject<List<Tran
27
28     }
29
30     protected override async Task OnMembersAddedAsync(IList<ChannelAcc
31     {
32         var welcomeText = "Hello and welcome!";
33         foreach (var member in membersAdded)
34         {
35             if (member.Id != turnContext.Activity.Recipient.Id)
36             {
37                 await turnContext.SendActivityAsync(MessageFactory.Tex
38             }
39         }
40     }
41
42
43 // Root myDeserializedClass = JsonConvert.DeserializeObject<Root>(myJs
44 public class DetectedLanguage
45 {
46     public string language { get; set; }
47     public double score { get; set; }
48 }
49
50 public class Translation
51 {
52     public string text { get; set; }
53     public string to { get; set; }
54 }
```

```
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:

```
1 // Copyright (c) Microsoft Corporation. All rights reserved.
2 // Licensed under the MIT License.
3 //
4 // Generated with Bot Builder V4 SDK Template for Visual Studio EchoBot v4
5
6 using Azure;
7 using Azure.AI.TextAnalytics;
8 using Microsoft.Bot.Builder;
9 using Microsoft.Bot.Schema;
10 using System;
11 using System.Collections.Generic;
12 using System.Net.Http;
13 using System.Threading;
14 using System.Threading.Tasks;
15
16 namespace EchoBot1.Bots
17 {
18     public class EchoBot : ActivityHandler
19     {
20         protected override async Task OnMessageActivityAsync(ITurnContext<
21         {
22             var response = MakeTranslator(turnContext.Activity.Text);
23             var replyText = $"原始語系: { response[0].detectedLanguage.la
24             replyText += $"日文: { response[0].translations[0].text} { S}
25             replyText += $"英文: { response[0].translations[1].text} { S}
26             replyText += $"韓文: { response[0].translations[2].text} { S}
27             replyText += $"GetSentiment: {GetSentiment(turnContext.Activit
28             await turnContext.SendActivityAsync(MessageFactory.Text(replyT
29         }
30
31         public static string GetSentiment(string text)
32         {
33             const string cogSvcKey = "798ff63e8c494c029dc2280dd5766347";
34             const string cogSvcEndpoint = "https://testcog20211107.cogniti
35
36             // Create client using endpoint and key
37             AzureKeyCredential credentials = new AzureKeyCredential(cogSvc
38             Uri endpoint = new Uri(cogSvcEndpoint);
39             var client = new TextAnalyticsClient(endpoint, credentials);
40
41             // Call the service to get the detected language
42             var result = client.AnalyzeSentiment(text, "zh");
43             return (result.Value.Sentiment.ToString());
44         }
45
46         static List<TranslatorResponse> MakeTranslator(string msg)
47         {
48             HttpClient client = new HttpClient();
49             string endpoint = "https://api.cognitive.microsofttranslator.c
50
51             client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
52             client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Region
53
54             var JsonBody = "[{'text':'" + msg + "'}]";
```

```
55         var content =
56             new StringContent(JsonBody, System.Text.Encoding.UTF8, "app
57         var response = client.PostAsync(endpoint, content).Result;
58         var ResponseJson = response.Content.ReadAsStringAsync().Result
59         return Newtonsoft.Json.JsonConvert.DeserializeObject<List<Tran
60
61     }
62
63     protected override async Task OnMembersAddedAsync(IList<ChannelAcc
64     {
65         var welcomeText = "Hello and welcome!";
66         foreach (var member in membersAdded)
67         {
68             if (member.Id != turnContext.Activity.Recipient.Id)
69             {
70                 await turnContext.SendActivityAsync(MessageFactory.Tex
71             }
72         }
73     }
74 }
75
76 // Root myDeserializedClass = JsonConvert.DeserializeObject<Root>(myJs
77 public class DetectedLanguage
78 {
79     public string language { get; set; }
80     public double score { get; set; }
81 }
82
83 public class Translation
84 {
85     public string text { get; set; }
86     public string to { get; set; }
87 }
88
89 public class TranslatorResponse
90 {
91     public DetectedLanguage detectedLanguage { get; set; }
92     public List<Translation> translations { get; set; }
93 }
94
95
96 }
97 }
```

## 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/) (<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

```
1  md folder
2  cd folder
3  dotnet new console
4  dotnet add package Microsoft.Extensions.Configuration.Json
5  dotnet add package Microsoft.Azure.CognitiveServices.Vision.CustomVision.P
6  code .
```

or 直接clone

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

## 3. 參考github程式

<https://github.com/isdaviddong/AI-102-CustomVision-classifyImage>  
<sub>(<https://github.com/isdaviddong/AI-102-CustomVision-classifyImage>)</sub>

## 4. 建立或修改 appsettings.json

注意

```
1  <Project Sdk="Microsoft.NET.Sdk">
2
3      <PropertyGroup>
4          <OutputType>Exe</OutputType>
5          <TargetFramework>net5.0</TargetFramework>
6      </PropertyGroup>
7      <!--注意這一塊-->
8      <ItemGroup>
9          <None Update="appsettings.json">
10             <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
11         </None>
12     </ItemGroup>
13     <!--注意-->
14     <ItemGroup>
15         <PackageReference Include="Microsoft.Azure.CognitiveServices.Vision.Cu
16             <PackageReference Include="Microsoft.Extensions.Configuration.Json" Ve
17         </ItemGroup>
18
19     </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

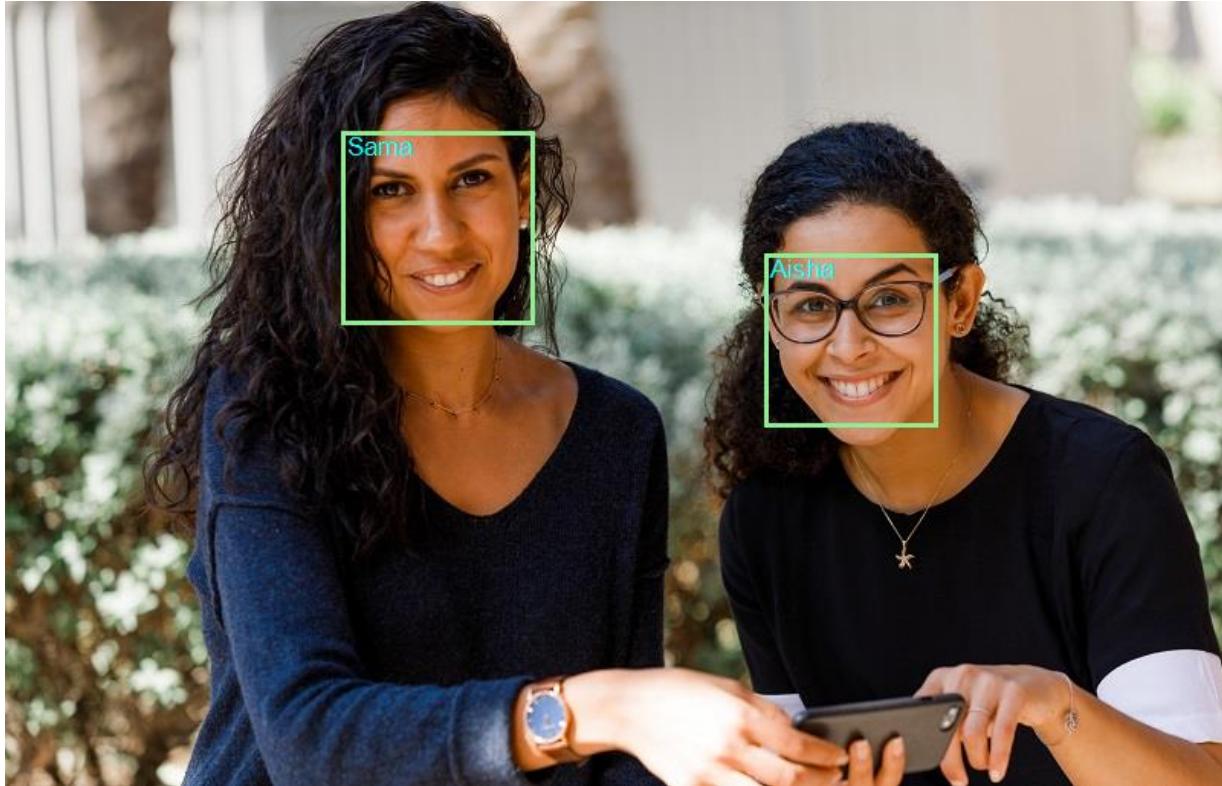
[\(https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/18-object-detection.html\)](https://microsoftlearning.github.io/AI-102-AIEngineer/Instructions/18-object-detection.html).

## 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

```

    檔案總管
    已開啟的檢視器
        appsettings.json
        Note.jpg
        images
    AI-102-LAB20-OCR
        vscode
        bin
        images
            Hand Writing.png
            Lincoln.jpg
        Note.jpg
        Rome.pdf
        obj
        appsettings.json
        ocr_results.jpg
        Program.cs
        read-text.csproj

```

Note.jpg

images > Note.jpg

AI-102-LAB20-OCR

終端機

```

Outdoors
Jun-Sep · Avg. 1.9 in precip · Avg. H 83°F · L 66°F
Book your trip now at www.margiestravel.com
PS D:\AI-102-Lab20-ocr> dotnet run
1: Use OCR API
2: Use Read API
3: Read handwriting
Any other key to quit
Enter a number:
3
Reading text in images/Note.jpg

Shopping List
Non-Fat milk
Bread
Eggs
PS D:\AI-102-Lab20-ocr> dotnet run

```

## 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)

```

1 POST /formrecognizer/v2.1/prebuilt/businessCard/analyze HTTP/1.1
2 Host: testcog20211107.cognitiveservices.azure.com
3 Ocp-Apim-Subscription-Key: 798ff63XXXXXc029dc2280dd5766347
4 locale: zh-hans
5 Content-Type: image/jpeg
6 Content-Length: 22
7
8 "<file contents here>"
```

The screenshot shows the Postman interface with a POST request to `https://testcog20211107.cognitiveservices.azure.com/formrecognizer/v2.1/prebuilt/businessCard/analyze`. The 'Headers' tab is selected, showing two key-value pairs: `Ocp-Apim-Subscription-Key` and `locale`. The response status bar indicates `Status: 202 Accepted`, `Time: 859 ms`, and `Size: 445 B`. A red arrow points to the 'Send' button at the top right, another points to the 'Headers' tab, and a third points to the status bar.

1. 辨識名片圖檔
2. 取得辨識結果 (get)

```

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

The screenshot shows a Postman request to the Azure Form Recognizer API. The method is GET, and the URL is <https://testcog20211107.cognitiveservices.azure.com/formrecognizer/v2.1/prebuilt/businessCard/analyzeResults/7cf3ce96-5...d74>. The response status is 200 OK. The response body is a JSON object representing a business card analysis. A specific website entry is highlighted with a red box:

```

    "websites": {
        "type": "array",
        "valueArray": [
            {
                "type": "string",
                "valueString": "www.isRock.com.tw",
                "text": "www.isRock.com.tw",
                "boundingBox": [
                    538,
                    397,
                    1439,
                    395,
                ]
            }
        ]
    }
  
```

## 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

### 恒逸問卷:

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

code : 211025

## 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](http://www.metricsthatmatter.com/ucominfo97) 網頁
- CISCO課程請進入[www.metricsthatmatter.com/ucom](http://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

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

Q 請輸入班別 :  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](http://www.metricsthatmatter.com/ucominfo97) 網頁
- CISCO課程請進入[www.metricsthatmatter.com/ucom](http://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



## 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