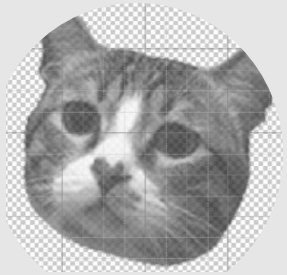
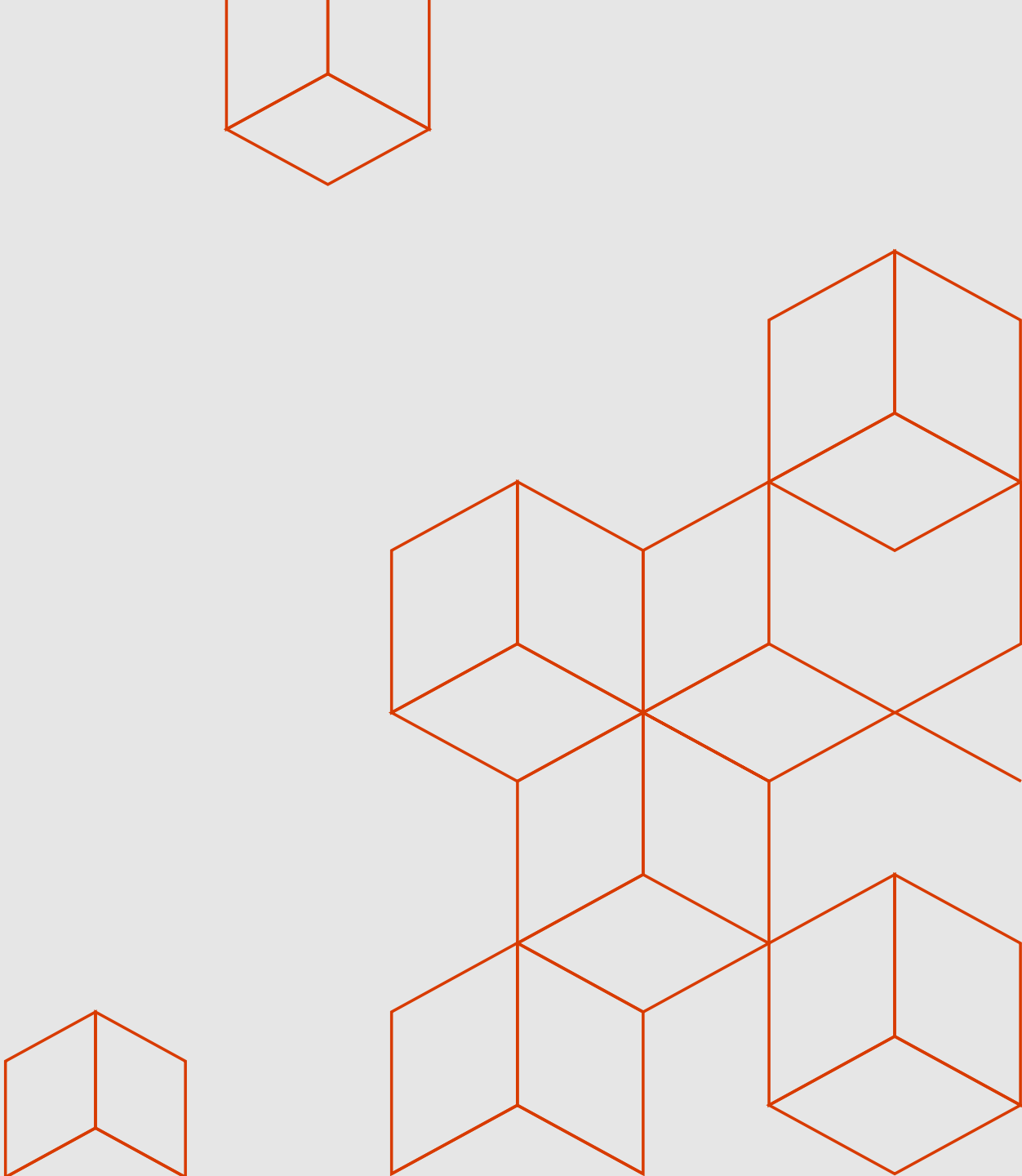
Microsoft

Azure Functions with Python



黃慧兒 / 微軟技術中心

Why Serverless?



有甚麼好處？

專注

解決商業邏輯問題
而非解決複雜煩悶的技術問題

效率

縮短上市時間
固定成本轉換為可變成本
更好的服務穩定性
更好的開發和測試管理
減少浪費

靈活

簡化起始體驗
更多的靈活性
更容易實驗
按照你的步伐擴展 - 不需要在第 1 天決定
原生適合微服務

專注於代碼，而不是硬體

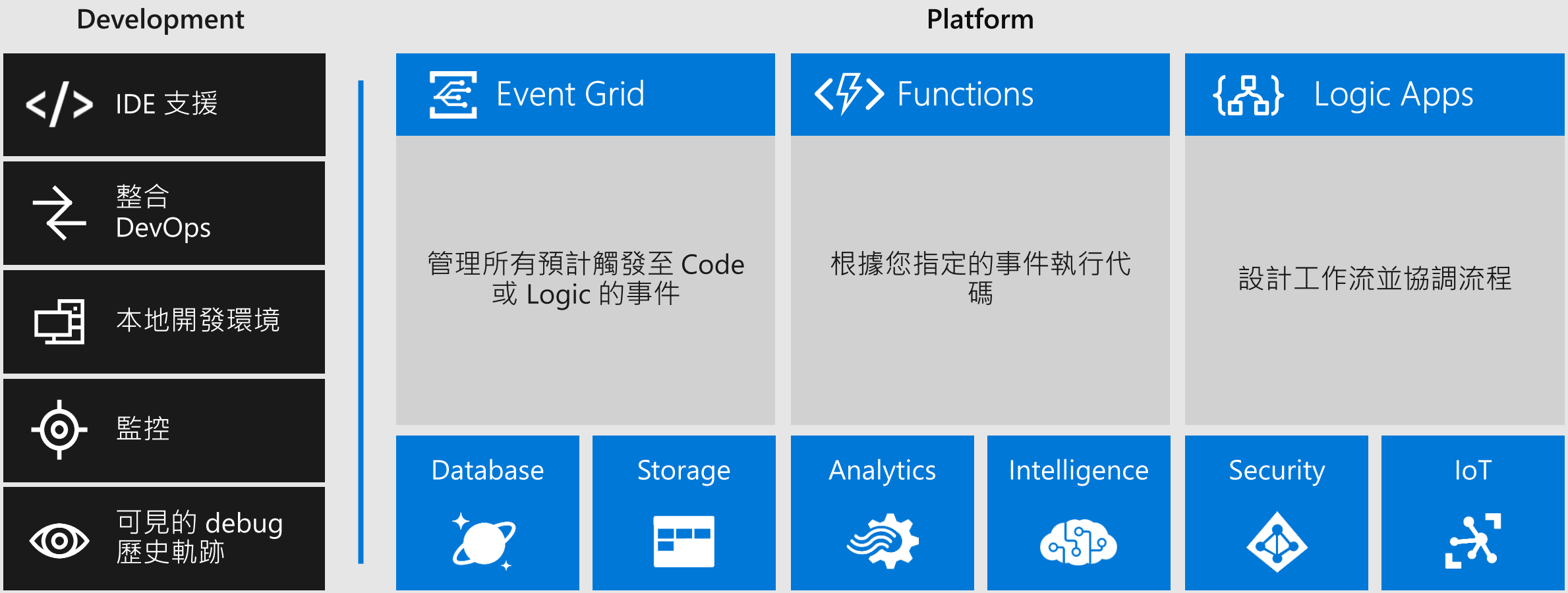
不需要維運環境管理

自動因應你的附載擴展

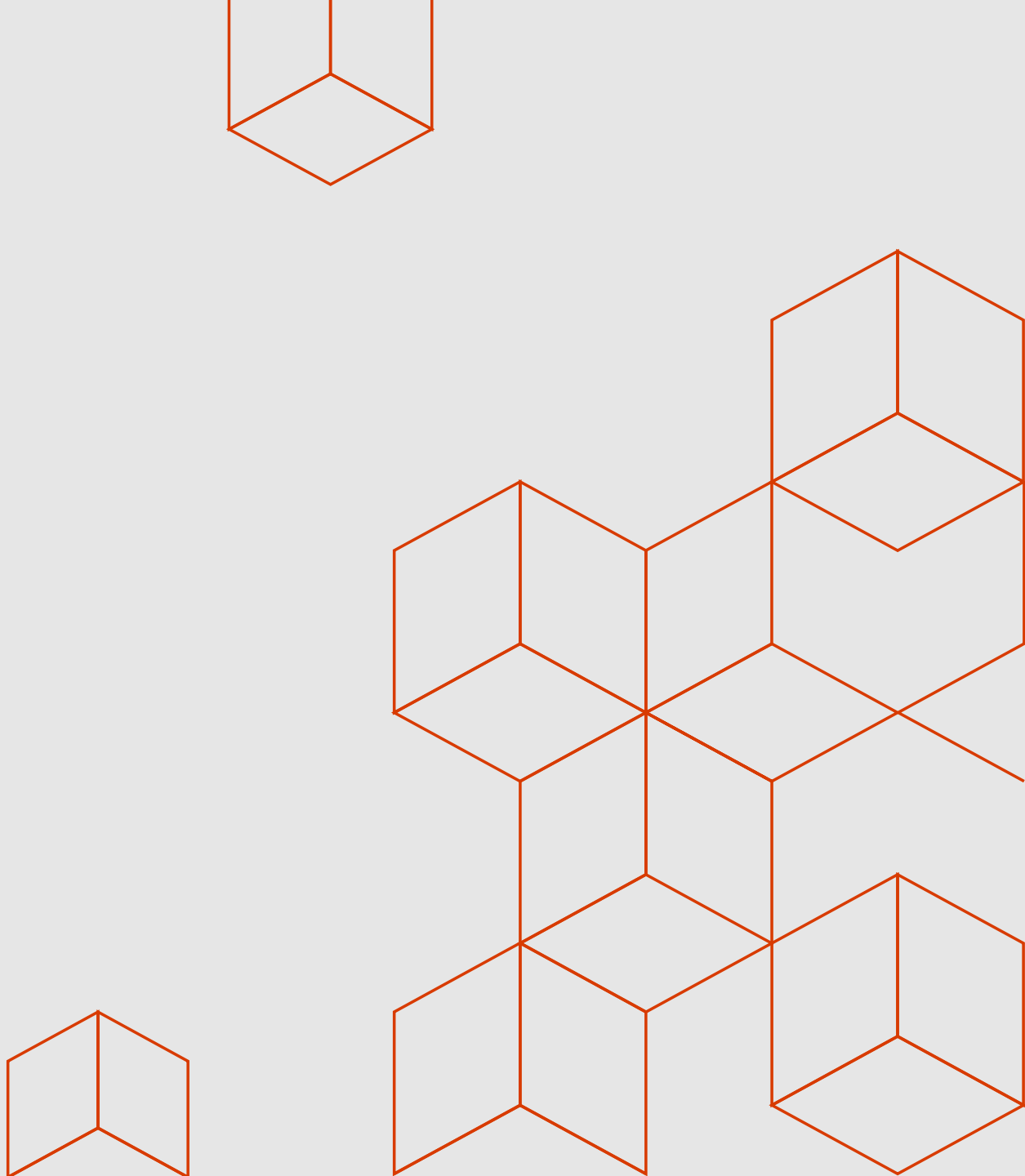
沒有浪費的資源
只為您使用的東西付費

Full integration with Azure ecosystem

Functions is the center piece of the Serverless platform



Azure Functions



What is Azure Function

An event-based, serverless compute experience that accelerates app development.

Event driven and scalable

Use built-in triggers and bindings to define when a function is invoked and to what data it connects.

Enhanced developer experience

Code, test and debug locally using your preferred editor or the easy-to-use web based interface including monitor

簡單來說

開發 / 偵錯

- 可以透過使用習慣使用的 Visual Studio Code 來建置 Function App

功能

- 觸發如 HTTP、Azure Storage 或是自己的訊息系統等的事件
- 處理訊息/資訊/資料到其他服務

部屬

- 在自己的電腦上客製化 Function Code，透過雲端服務來做彈性的擴充
- 整合 Azure DevOps 做 CI/CD

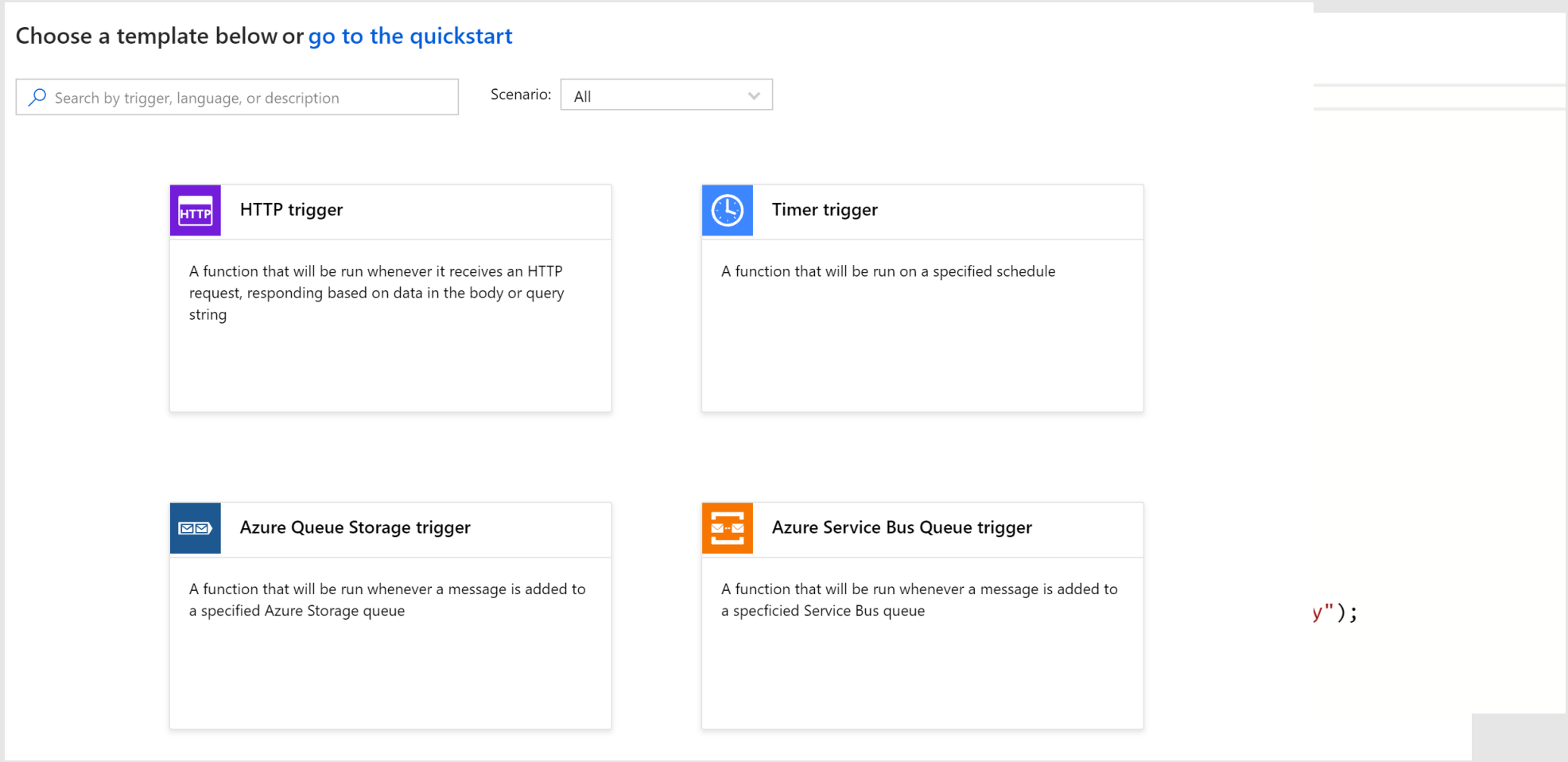
開發 / 偵錯

- 使用 Azure Portal
  - 僅能在 **Window 函式**上使用
  - 如果要客製開發，不建議直接在雲端上開發
- 使用 Azure Function Core Tool
- 使用 Visual Studio Code
- 使用 Visual Studio

Azure Portal

如果使用 **Windows 函式**，可以直接在 Azure Portal 上面做設定

從 新增函式 > 定義函式的 Binding > 撰寫程式碼 > 直接在雲端上執行



Azure Function Core Tool

Azure Functions Core Tools 有兩個版本。 您使用的版本取決於您的本機開發環境、選擇的語言，以及所需的支援層級：

Version 1.x

- **僅在 Windows 電腦上提供支援**
- 從 npm 套件進行安裝

Version 2.x

- **支援 Windows、macOS 和 Linux**。
- 使用平台專屬的套件管理員或 npm 進行安裝。
- 以 .NET Core 為建置基礎，.NET Core 2.x 支援的所有平台都支援此版本

Azure Function Core Tool

建立新專案

1) func init *YOURPROJNAME*

```
λ func init PythonFuncProj
```

2) 選擇 Python

```
Select a worker runtime:
dotnet
node
python
powershell (preview)
```

3) 完成

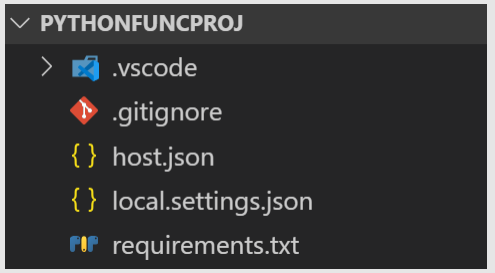
```
Select a worker runtime: python
Writing .gitignore
Writing host.json
Writing local.settings.json
Writing C:\Users\huihuan\source\repos\PythonFuncProj\.vscode\extensions.json
```

Azure Function Core Tool

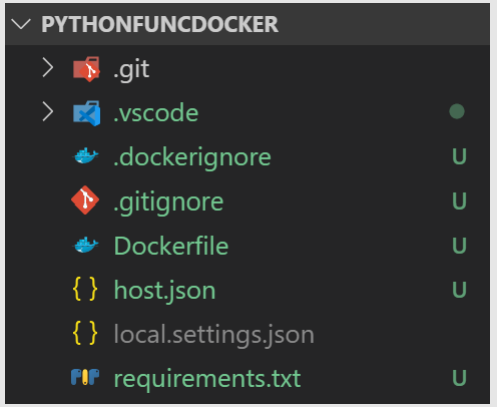
```
λ func init PythonFuncDocker --docker --source-control --force
```

--docker	使用選擇的 worker run time 為基礎的基底映像建立 Dockerfile
--force	以相同名稱覆寫現有同檔名的專案
--source-control	控制是否要建立 Git 存放庫。 2.x 依預設不會建立存放庫。 設為 true 時，會建立存放庫
--worker-runtime	設定專案的語言執行階段。 支援的值為 dotnet、node (JavaScript) java 和 python。 未設定此選項時，系統會在初始化期間提示您選擇執行階段。

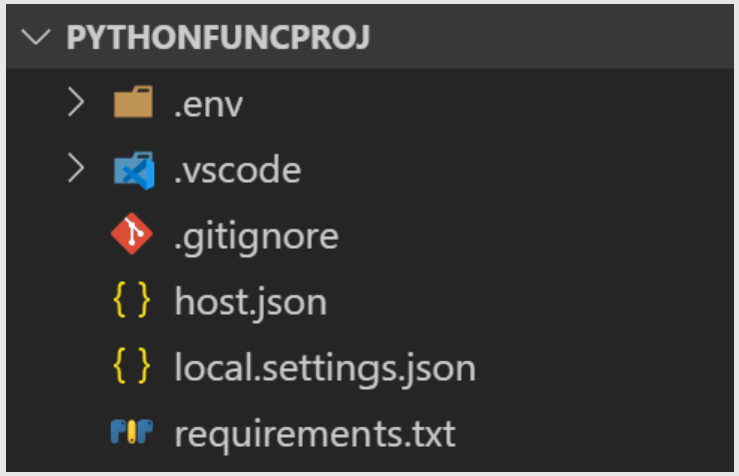
未使用



有使用



專案結構



.env

- 使用 virtualenv 建立 python 虛擬環境

.gitignore

- 紀錄git push時不會上傳的檔案

host.json

- 在檔案中註冊 Function 中使用的 trigger 類型的擴充功能。
- 若使用的是 HTTP trigger 和計時器觸發程序，則不需要延伸模組

local.setting.json (本機設定檔)

- 儲存應用程式設定、連接字串及本機開發工具所使用的設定。 僅在本機執行專案時才會使用本機 .settings. json 檔案中的設定
- 上傳專案到雲端時不會上傳此份檔案
- 透過 python 取得環境變數的方式 os.environ 取得檔案的值

requirements.txt

- 紀錄專案所需要的套件，透過 pip 來做安裝

Bindings and integrations

Functions 2.0

Microsoft.NET.Sdk.Functions (.NET Standard 2.0)

- HTTP
- Timer

Microsoft.Azure.WebJobs.Extensions.Storage 3.0.0

Microsoft.Azure.WebJobs.Extensions.ServiceBus 3.0.0

Microsoft.Azure.Webjobs.Extensions.EventHubs 3.0.0

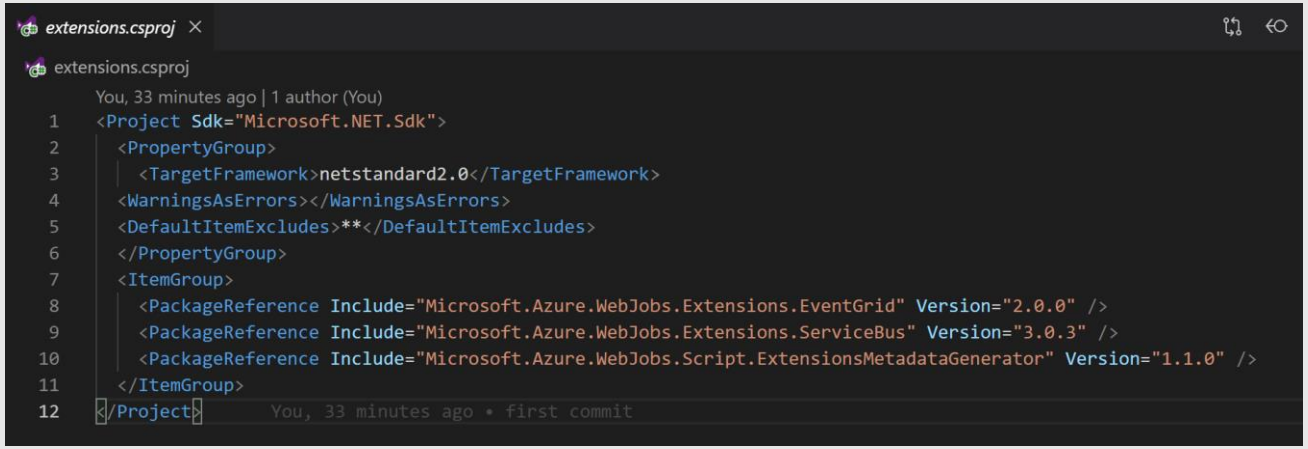
Microsoft.Azure.WebJobs.Extensions.CosmosDB 3.0.0

Microsoft.Azure.Webjobs.Extensions.EventGrid 2.0.0

Microsoft.Azure.Webjobs.Extensions.MicrosoftGraph 1.0.0-beta6

Microsoft.Azure.WebJobs.Extensions.DurableTask 1.4.0

Microsoft.Azure.Webjobs.Extensions.SignalRService 1.0.0-preview1-10002



功能

Functions 適用於處理資料、整合系統、IoT，及建置簡單 API 和微服務等場景

HTTPTrigge	預先定義的排程執行清除或其他批次工作
TimerTrigger	預先定義的排程執行清除或其他批次工作
CosmosDBTrigger	當 Azure Cosmos DB document 在 NoSQL 資料庫的 collection 中新增或更新時觸發
BlobTrigger	在新增至 Blob 時，處理 Azure 儲存體 blob
QueueTrigger	在訊息送達 Azure 儲存體佇列中時回應
EventGridTrigger	回應傳遞到 Azure Event Grid 之訂用帳戶的事件。適合用於建置事件導向的架構
EventHubTrigger	回應傳送到 Azure Event Hub 的事件，適用於應用程式檢測、使用者經驗或工作流程處理及物聯網 (IoT) 案例
ServiceBusQueueTrigger	透過監聽 message queue，將程式碼連接至其他 Azure 服務或on-premises的服務
ServiceBusTopicTrigger	透過訂閱topic，將程式碼連接至其他 Azure 服務或on-premises的服務

Azure Functions

Events

React to timers, HTTP, or events from your favorite Azure services, with more on the way

Code

Author functions in C#, Node.JS, Java, Python and more

Outputs

Send results to an ever-growing collection of services

Azure Function Core Tool

為專案建立新的 function (函式)

1) 選擇要使用的事件功能

```
(.env) λ func new
Select a template:
Azure Blob Storage trigger
Azure Cosmos DB trigger
Azure Event Grid trigger
Azure Event Hub trigger
HTTP trigger
Azure Queue Storage trigger
Azure Service Bus Queue trigger
Azure Service Bus Topic trigger
Timer trigger
```

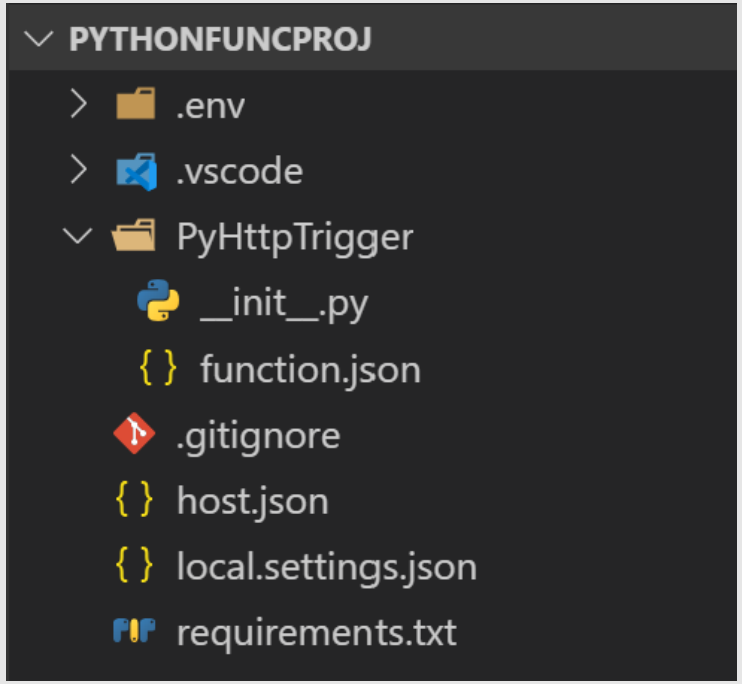
2) 輸入function (函式) 名稱

```
Function name: [HttpTrigger]
```

3) 完成

```
Writing C:\Users\huihuan\source\repos\PythonFuncProj\PyHttpTrigger\__init__.py
Writing C:\Users\huihuan\source\repos\PythonFuncProj\PyHttpTrigger\function.json
The function "PyHttpTrigger" was created successfully from the "HTTP trigger" template.
```

專案結構



PyHttpTrigger

- 特定功能的 function (函式)資料夾

__init__.py

- 函式實際執行的程式碼

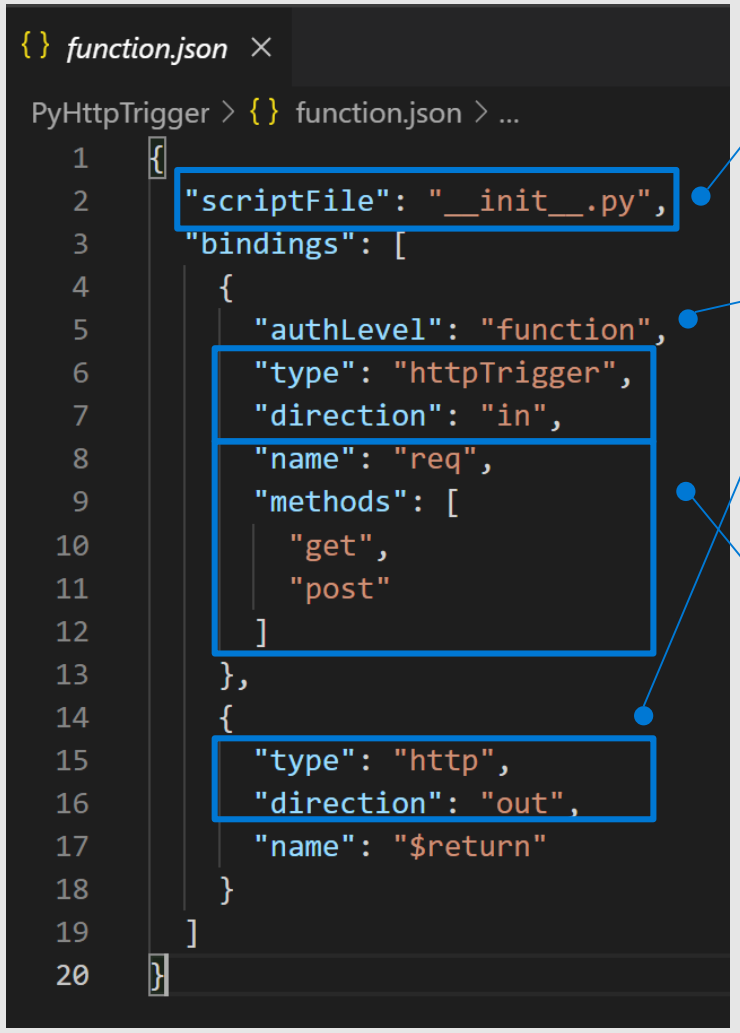
function.json

- 定義 functions(函式)的 Binding 及其他基本的組態設定
- 如何trigger這個函式，以及輸出的類型

各功能具體實作

HTTP Trigger

function.json



一開始的執行的檔案，可以修改scriptFile的值來更改為其他檔案

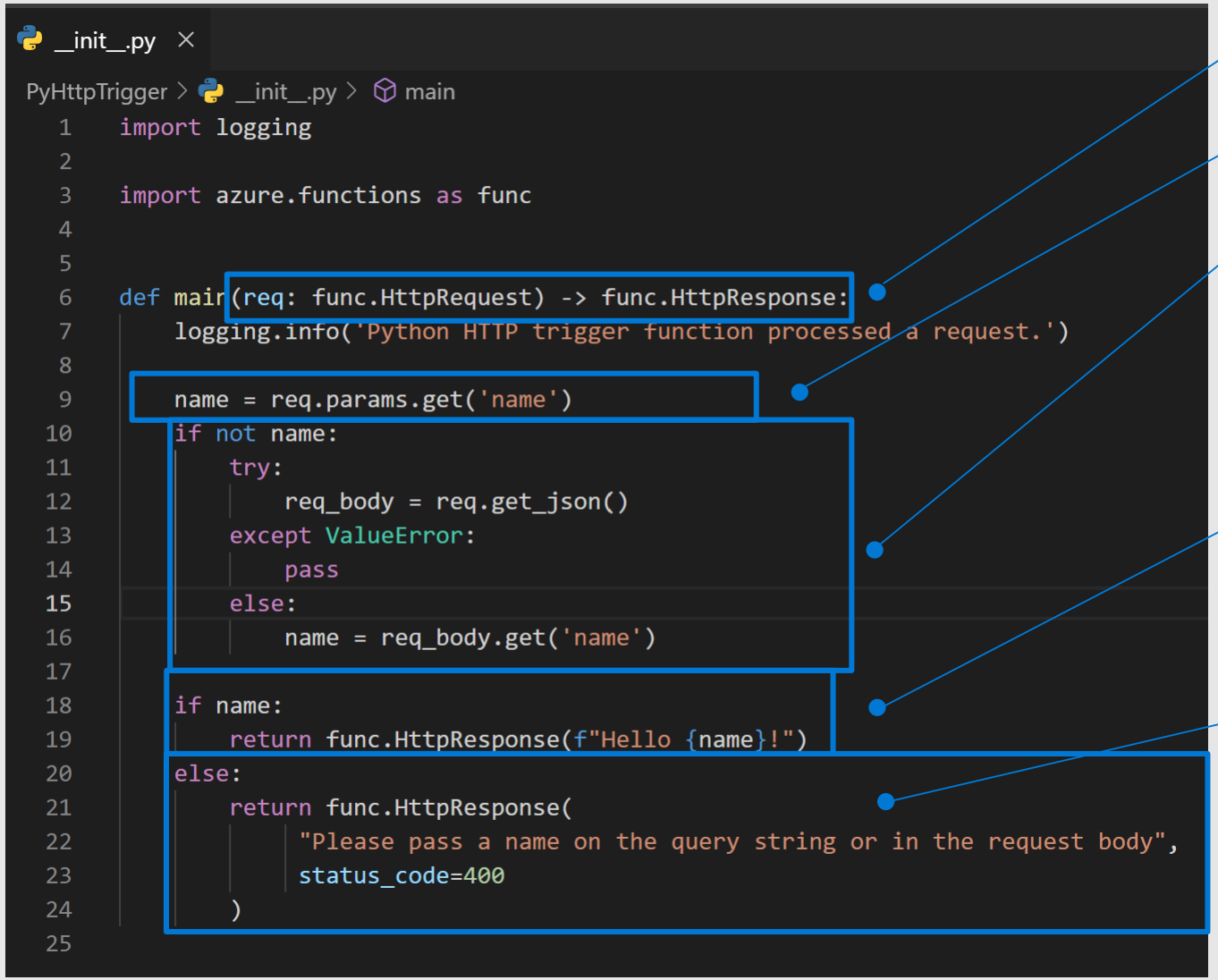
- 可以額外增加 entryPoint 屬性來指定進入點

所有觸發程序和繫結在 function.json 檔案中都具有 direction 屬性：

- 對於觸發程序，方向一律為 in
- 輸入和輸出繫結使用 in 和 out

- 透過 HTTP request 來 trigger 此函式，並將此方法命名為 req
- 這裡的 req 會對應到 __init__.py 來做使用，會使用到 get 及 post

__init__.py



- 宣告輸入及輸出的型態
- 用 GET 方式取得參數name的值
- 若 name 為 false (name 為空值)
  - 則透過 get_json 來獲取數據並儲存為 dictionary 的格式並命名為 req_body
  - 從 req_body 取得參數 name 的值
- 若 name 為 true (name 不為空值)
  - 回傳 HttpResponse “Hello {name}”
    (在頁面上顯示使用者輸入name的值)
- 其他狀況
  - 回傳 HttpResponse “Please pass a name ...” 及status code 顯示為 400
    (在頁面上顯示要求使用者輸入值)

HTTP Trigger

1) 在專案的根目錄下使用 *func host start* 將函式在本機跑起來

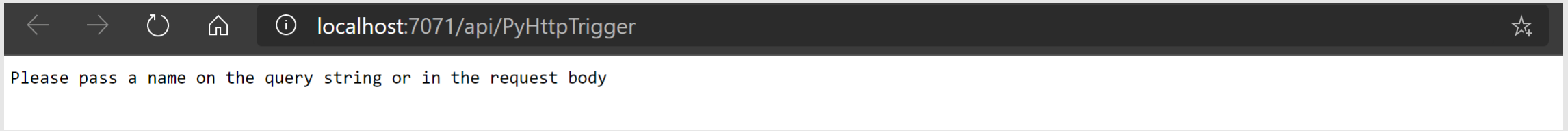
```
(.env) λ func host start
```

2) 連結到 http//localhost:7071/api/PyHttpTrigger

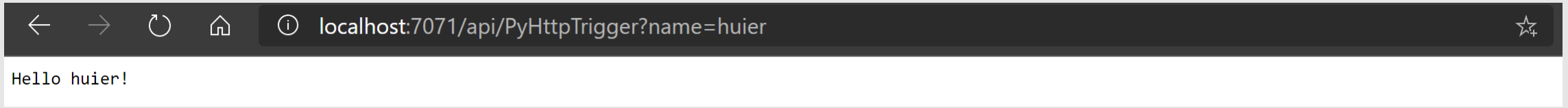
```
[10/1/2019 8:43:49 PM] Job host started
Hosting environment: Production
Content root path: C:\Users\huihuan\source\repos\PythonFuncProj
Now listening on: http://0.0.0.0:7071
Application started. Press Ctrl+C to shut down.

Http Functions:

        PyHttpTrigger: [GET,POST] http://localhost:7071/api/PyHttpTrigger

[10/1/2019 8:43:49 PM]  INFO: Starting Azure Functions Python Worker.
[10/1/2019 8:43:49 PM]  INFO: Worker ID: 91567091-a97c-4b57-a8e7-53248a6b9537, Request ID: eb8e6262-7888-4caf-b446-2f241a09de6d, Host Addr
ess: 127.0.0.1:5791
[10/1/2019 8:43:49 PM]  INFO: Successfully opened gRPC channel to 127.0.0.1:5791
[10/1/2019 8:43:49 PM]  INFO: Received WorkerInitRequest, request ID eb8e6262-7888-4caf-b446-2f241a09de6d
[10/1/2019 8:43:49 PM]  INFO: Received FunctionLoadRequest, request ID: eb8e6262-7888-4caf-b446-2f241a09de6d, function ID: ab5f97c4-92a4-4
bbc-b253-3cead987adaf
[10/1/2019 8:43:49 PM]  INFO: Successfully processed FunctionLoadRequest, request ID: eb8e6262-7888-4caf-b446-2f241a09de6d, function ID: a
b5f97c4-92a4-4bbc-b253-3cead987adaf
[10/1/2019 8:43:54 PM] Host lock lease acquired by instance ID '00000000000000000000000CE98A476'.
```

HTTP Trigger

3) 透過發送 http request 來啟動 function 成功！



4) 輸入參數的值



5) 使用 *func host start --port 8080* 可以改變 port

```
[10/1/2019 8:55:01 PM] Host initialized (399ms)
[10/1/2019 8:55:01 PM] Host started (447ms)
[10/1/2019 8:55:01 PM] Job host started
Hosting environment: Production
Content root path: C:\Users\huihuan\source\repos\PythonFuncProj
Now listening on: http://0.0.0.0:8080
Application started. Press Ctrl+C to shut down.

Http Functions:

        PyHttpTrigger: [GET,POST] http://localhost:8080/api/PyHttpTrigger
```

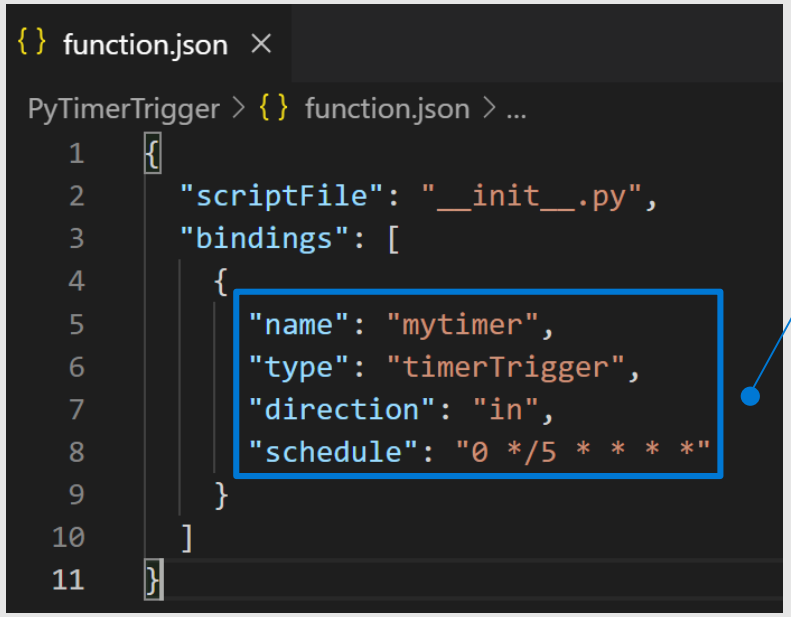
HTTP Trigger

5) 最後，你可以透過本機的 console 來做偵錯的動作

```
[10/1/2019 8:50:53 PM] Executing 'Functions.PyHttpTrigger' (Reason='This function was programmatically called via the host APIs.', Id=eb12
785e-6592-48a6-aa2a-34bde38ef6af)
[10/1/2019 8:50:53 PM]  INFO: Received FunctionInvocationRequest, request ID: eb8e6262-7888-4caf-b446-2f241a09de6d, function ID: ab5f97c4-
92a4-4bbc-b253-3cead987adaf, invocation ID: eb12785e-6592-48a6-aa2a-34bde38ef6af
[10/1/2019 8:50:53 PM] Python HTTP trigger function processed a request.
[10/1/2019 8:50:53 PM]  INFO: Successfully processed FunctionInvocationRequest, request ID: eb8e6262-7888-4caf-b446-2f241a09de6d, function
 ID: ab5f97c4-92a4-4bbc-b253-3cead987adaf, invocation ID: eb12785e-6592-48a6-aa2a-34bde38ef6af
[10/1/2019 8:50:53 PM] Executed 'Functions.PyHttpTrigger' (Succeeded, Id=eb12785e-6592-48a6-aa2a-34bde38ef6af)
[10/1/2019 8:50:53 PM] Executed HTTP request: {
[10/1/2019 8:50:53 PM]   "requestId": "b1bdade1-bcd6-4acb-a5fa-209ac0afb0d4",
[10/1/2019 8:50:53 PM]   "method": "GET",
[10/1/2019 8:50:53 PM]   "uri": "/api/PyHttpTrigger",
[10/1/2019 8:50:53 PM]   "identities": [
[10/1/2019 8:50:53 PM]     {
[10/1/2019 8:50:53 PM]       "type": "WebJobsAuthLevel",
[10/1/2019 8:50:53 PM]       "level": "Admin"
[10/1/2019 8:50:53 PM]     }
[10/1/2019 8:50:54 PM]   ],
[10/1/2019 8:50:54 PM]   "status": 200,
[10/1/2019 8:50:54 PM]   "duration": 343
[10/1/2019 8:50:54 PM] }
```

Timer Trigger

function.json



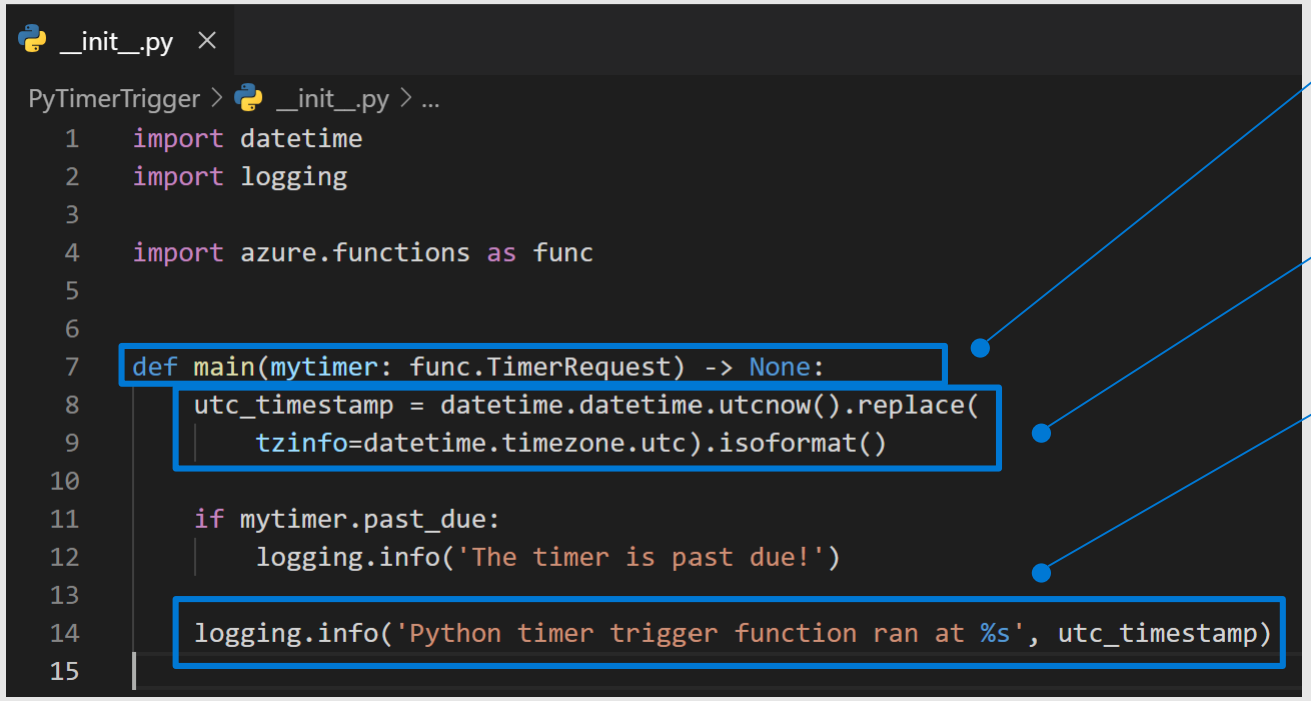
- 每隔五分鐘觸發一次 Timer Trigger：
  - 使用 NCRONTAB 運算式，類似於 CRON 運算式
  - 不同之處在於它會在開頭包含一個額外的第六個欄位，以供時間精確度 (以秒為單位) 使用

```
{second} {minute} {hour} {day} {month} {day-of-week}
```

  - CronTab 命令常見於 Unix 和 類Unix 的作業系統之中，用於設定週期性被執行的指令
  - 如果覺得不太會計算，線上都有工具可以使用

    https://crontab.guru/

__init__.py



- 傳入 TimerRequest 啟動程式，沒有output
- 紀錄目前的時間
- 在 console 印出字串及目前時間

Timer Trigger

1) 在專案的根目錄下使用 *func host start* 將函式在本機跑起來

```
(.env) λ func host start
```

2) 每五分鐘觸發一次

```
[10/2/2019 4:57:29 AM] Executing 'Functions.PyTimerTrigger' (Reason='Timer fired at 2019-10-02T12:57:29.9994831+08:00', Id=062fa572-9b71-4dec-a996-48d7fa730f06)
[10/2/2019 4:57:30 AM]  INFO: Received FunctionInvocationRequest, request ID: 0b633568-a304-4fab-8934-9ceb75c97b55, function ID: 833b3a25-11c8-44a9-a9a8-ec51c514fb64, invocation ID: 062fa572-9b71-4dec-a996-48d7fa730f06
[10/2/2019 4:57:30 AM]  INFO: Successfully processed FunctionInvocationRequest, request ID: 0b633568-a304-4fab-8934-9ceb75c97b55, function ID: 833b3a25-11c8-44a9-a9a8-ec51c514fb64, invocation ID: 062fa572-9b71-4dec-a996-48d7fa730f06
[10/2/2019 4:57:30 AM] Python timer trigger function ran at 2019-10-02T04:57:30.002369+00:00
[10/2/2019 4:57:30 AM] Executed 'Functions.PyTimerTrigger' (Succeeded, Id=062fa572-9b71-4dec-a996-48d7fa730f06)
```

Timer Trigger - 需要注意

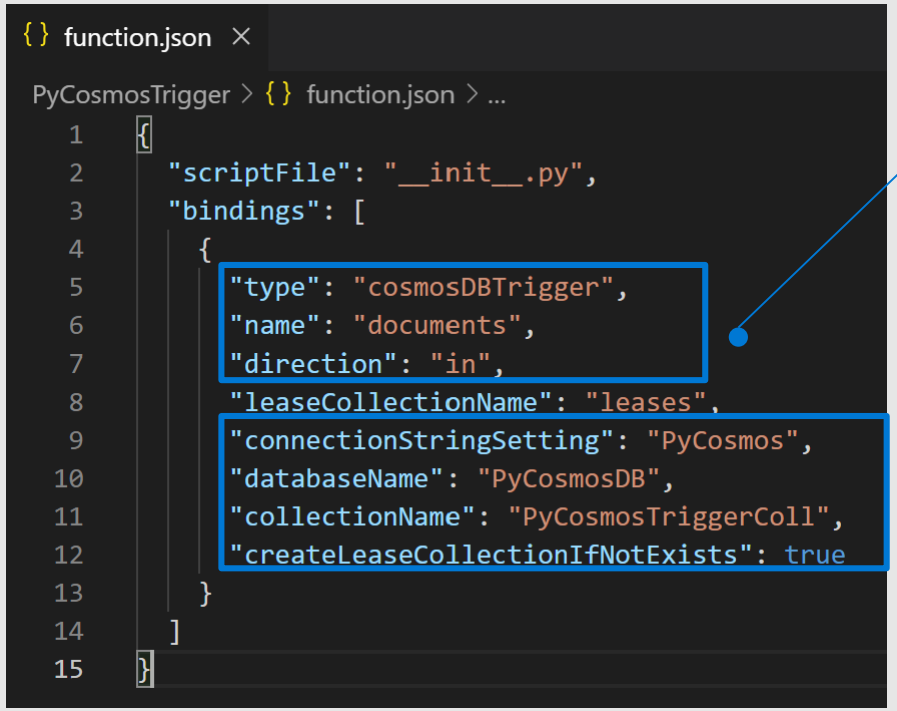
- 顯示 Microsoft.WindowsAzure.Stroage 設定有問題

```
[10/2/2019 4:43:06 AM] Host initialized (325ms)
[10/2/2019 4:43:06 AM] The listener for function 'Functions.PyTimerTrigger' was unable to start.
[10/2/2019 4:43:06 AM] The listener for function 'Functions.PyTimerTrigger' was unable to start. Microsoft.WindowsAzure.Storage: Settings must be of the form "name=value".
```

> 解決方法：修改 local.settings.json 中 AzureWebJobsStorage 的值改成Azure Storage Account 的連線字串

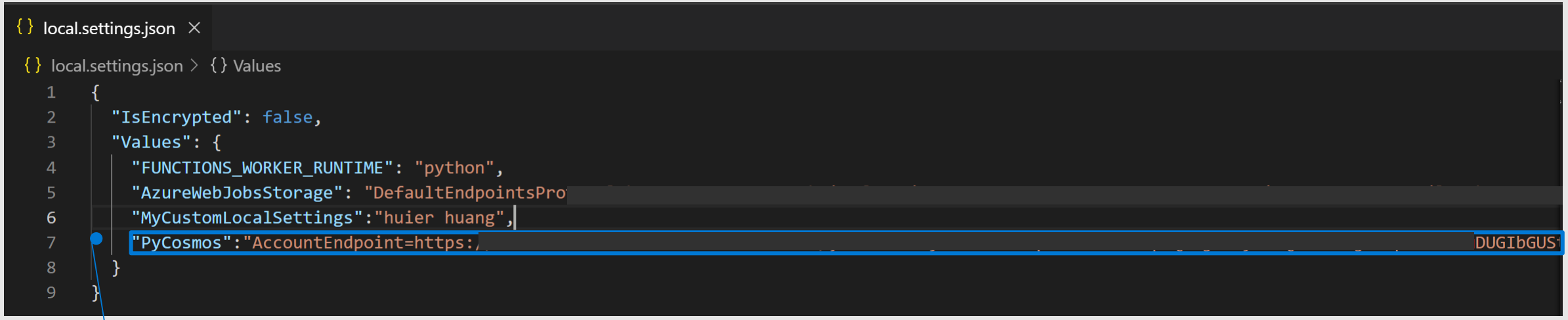
Cosmos Trigger

function.json



- 以Cosmos 的更新與否來Trigger函式的執行
- 設定 Cosmos 的連線資訊
  - ConnectionStringSetting : Azure Cosmos 的連線字串
  - databaseName : Cosmos D B中的 Data Base 名稱
  - collection：CosmosDB 中 Database 下的 Collection 名稱
  - 如果 collection 不存在則建立一個

Local.settings.json



必須將 Cosmos 的連線字串加在 local.setting.json 中，再帶到 Cosmos 函式的 function.json中

- 連線字串可以透過在 Azure Storage Exploer 中對資源按右鍵取得，或在 Azure Portal 上
- 不可直接將連線字串寫在 function.json中，會爆掉

__init__.py

__init__.py

PyCosmosTrigger > __init__.py > ...

```python
import logging

import azure.functions as func


def main(documents: func.DocumentList) -> str:
    if documents:
        logging.info('********* Cosmos have been update!! Document id: %s', documents[0]['id'])
```

透過 Cosmos document 的更新來啟動函式，傳入document的內容

- 若 document 非空值則打印出訊息及 document 的 id 欄位的值

Cosmos Trigger

1) 在專案的根目錄下使用 *func host start* 將函式在本機跑起來

```
(.env) λ func host start
```

2) Cosmos 中的 document 更新便執行打印出結果

```
[10/2/2019 8:49:46 AM] Executing 'Functions.PyCosmosTrigger' (Reason='New changes on collection PyCosmosTriggerColl at 2019-10-02T08:49:46.8634294Z', Id=2c1c81e7-7af4-47f2-b763-13a42bb5d082)
[10/2/2019 8:49:46 AM]  INFO: Received FunctionInvocationRequest, request ID: 6b8eeb8d-fb2c-4a0e-a280-083cc333508f, function ID: 2916902c-662d-4230-8091-04c47c2d47b5, invocation ID: 2c1c81e7-7af4-47f2-b763-13a42bb5d082
[10/2/2019 8:49:46 AM] ********* Cosmos have been update!! Document id: replace_with_new_document_id
[10/2/2019 8:49:46 AM]  INFO: Successfully processed FunctionInvocationRequest, request ID: 6b8eeb8d-fb2c-4a0e-a280-083cc333508f, function ID: 2916902c-662d-4230-8091-04c47c2d47b5, invocation ID: 2c1c81e7-7af4-47f2-b763-13a42bb5d082
[10/2/2019 8:49:46 AM] Executed 'Functions.PyCosmosTrigger' (Succeeded, Id=2c1c81e7-7af4-47f2-b763-13a42bb5d082)
```

Cosmos Trigger

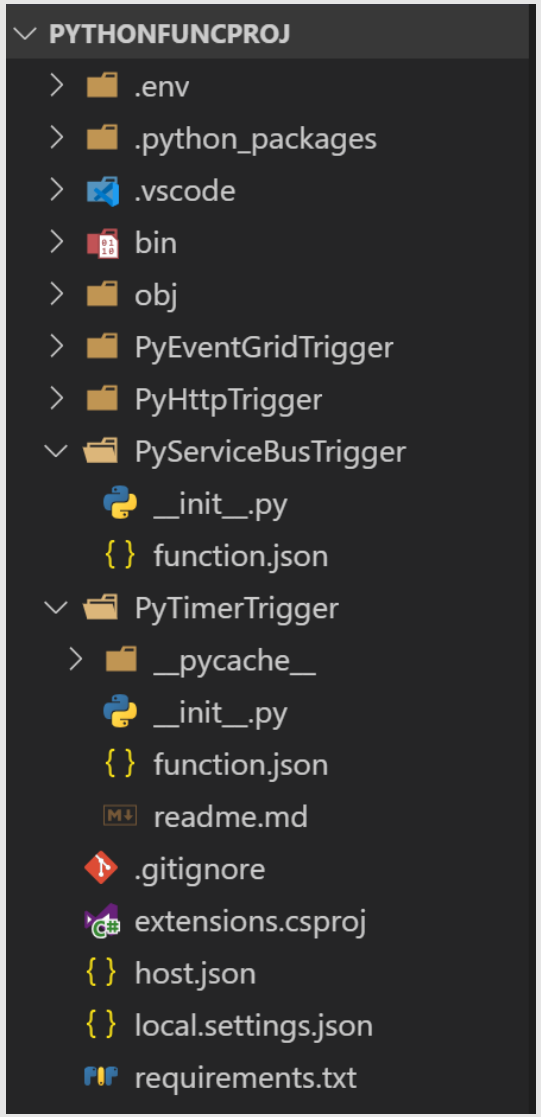
- 顯示 Microsoft.WindowsAzure.Stroage 設定有問題

```
[10/2/2019 4:43:06 AM] Host initialized (325ms)
[10/2/2019 4:43:06 AM] The listener for function 'Functions.PyTimerTrigger' was unable to start.
[10/2/2019 4:43:06 AM] The listener for function 'Functions.PyTimerTrigger' was unable to start. Microsoft.WindowsAzure.Storage: Settings must be of the form "name=value".
```

> 解決方法：修改 local.settings.json 中 AzureWebJobsStorage 的值改成Azure Storage Account 的連線字串

我可以把上面的函式都放在同一個函式 project 嗎？

專案結構



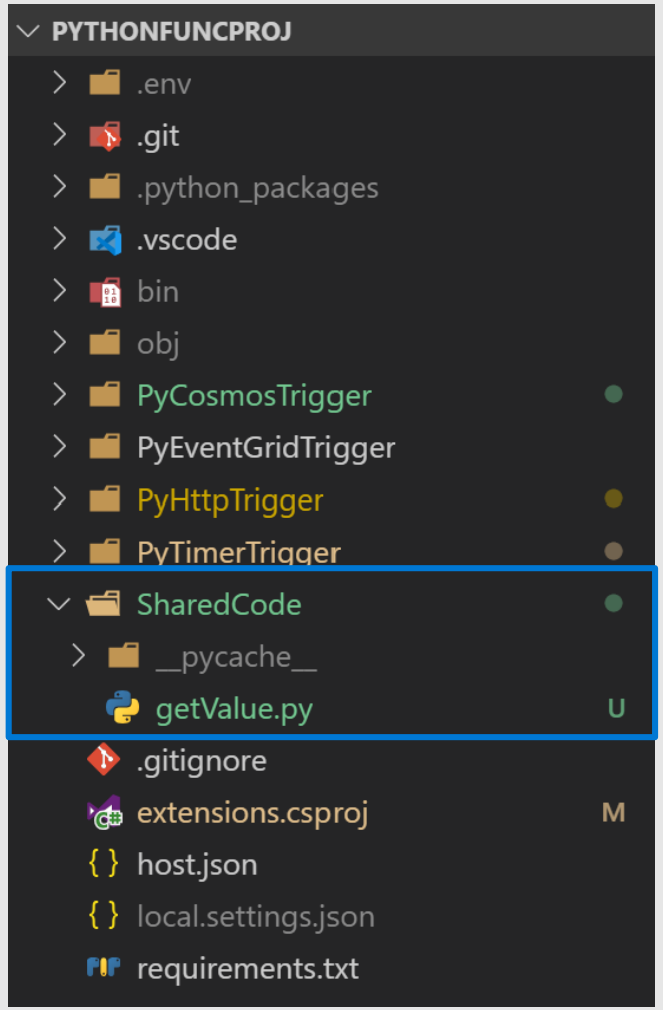
可以 (只是後面介紹的 real case 我是選擇分開放)

- 每個不同功能的函式會有自己的 __init__.py 來撰寫要執行的動作和 function.json 來描述 input/output
- 函式共用 host.json、local.settings.json 和requirements.txt

如果我有一些檔案是要共用的，要怎麼做？

專案結構

- 新增一個 SharedCode 資料夾



./PyHttpTrigger/__init__.py

PyTimerTrigger > __init__.py > ...

You, a few seconds ago | 1 author (You)

```python
import datetime
import logging

import azure.functions as func
from SharedCode import getValue


def main(mytimer: func.TimerRequest) -> None:
    utc_timestamp = datetime.datetime.utcnow().replace(
        tzinfo=datetime.timezone.utc).isoformat()
```

./PyTimerTrigger/__init__.py

PyHttpTrigger > __init__.py > ...

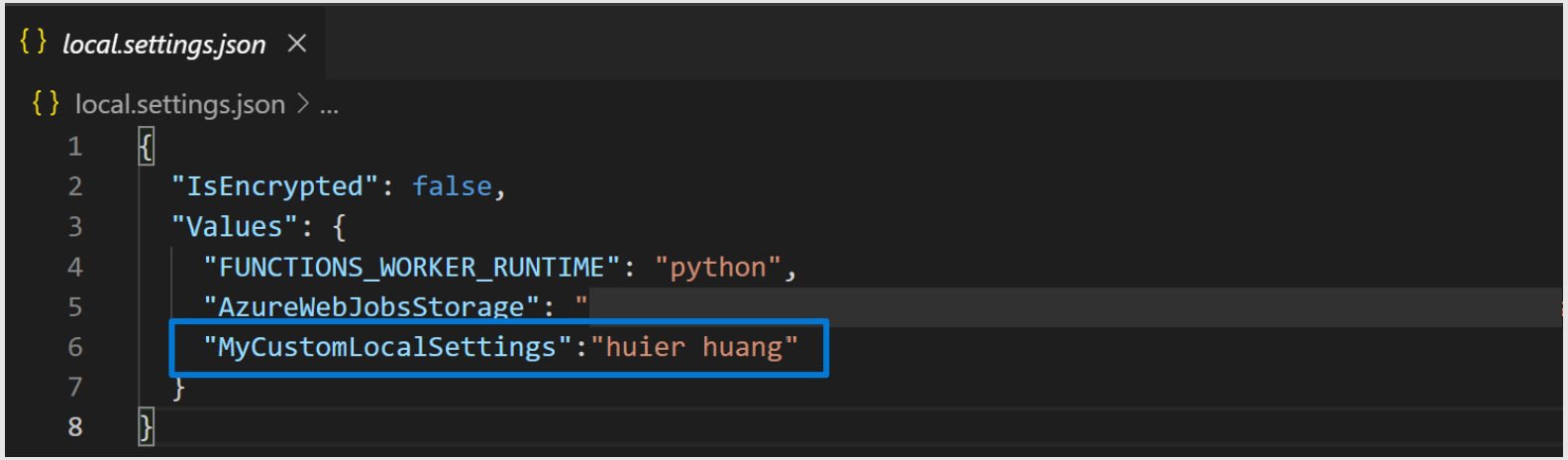
You, a minute ago | 1 author (You)

```python
import logging
import os
import azure.functions as func
from SharedCode import getValue


def main(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')

    name = req.params.get('name')
    if not name:
        try:
```

在程式碼中我有需要使用到一些連線字串等的設定，我該怎麼做？

環境變數

- 將連線字串等資訊寫在local setting中



- 程式碼中使用 os.environ[] 來取得環境變數

```
import logging
import os
import azure.functions as func
```

```
else:
    env = os.environ["MyCustomLocalSettings"]
    return func.HttpResponse(
         "The Value from Local Setting : {}".format(env),
        #  "Please pass a name on the query string or in the request body",
         status_code=400
    )
```

目前都在本機做 debug，如果 upload 到 Azure 上我要怎麼 debug

偵錯

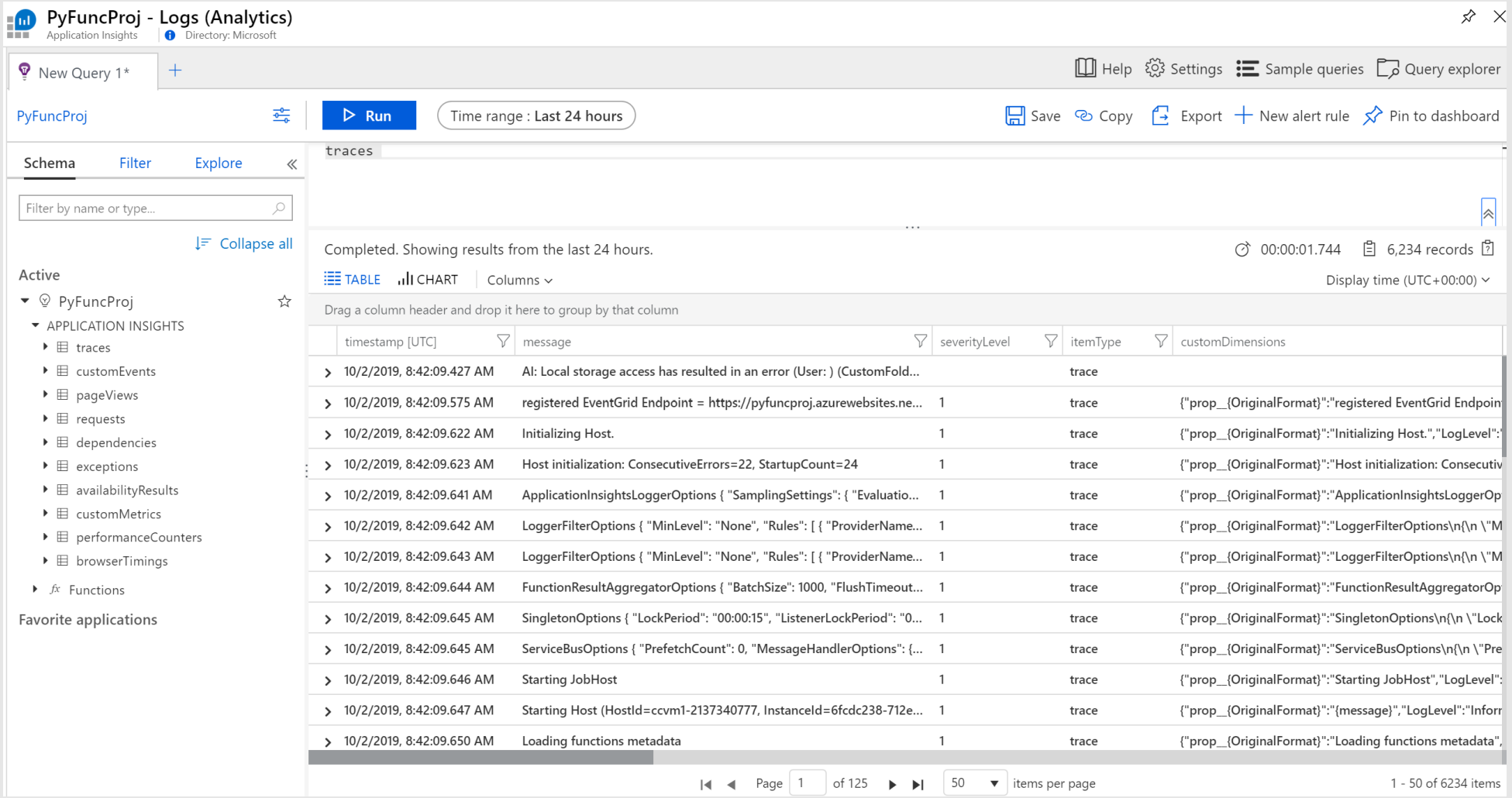
- 透過 python 中 logging 的方法來記錄相關資訊

```python
def main(documents: func.DocumentList) -> str:
    if documents:
        logging.info('********* Cosmos have been update!! Document id: %s', documents[0]['id'])
```

```python
if mytimer.past_due:
    logging.info('The timer is past due!')

logging.info('Python timer trigger function ran at %s', utc_timestamp)
```

偵錯

- 使用 Application Insight 蒐集 log，透過 Log Analytics query 出有紀錄在logging中的資訊

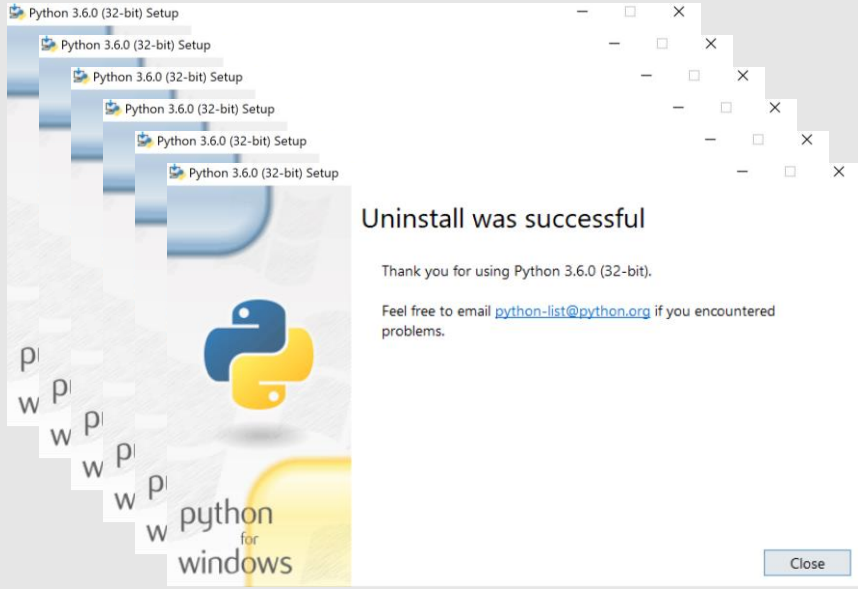


Azure Function with Python 之路充滿荊棘

需要注意

- 請安裝 **Python 3.6.x** 版本

> 解決方法：請重新安裝 (文件都寫了，為什麼不看！)



必要條件

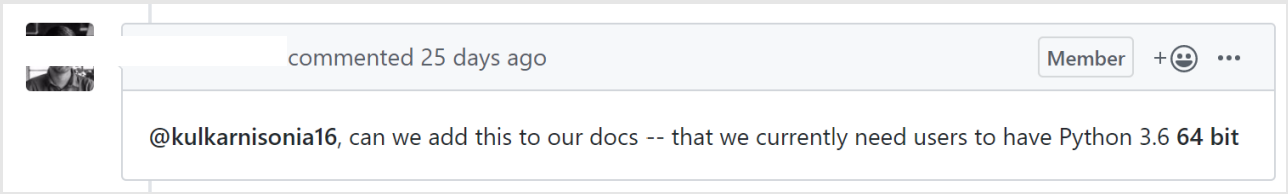
開始之前，您必須符合下列條件：

- 安裝 Python 3.6.x。

- 請安裝 **64 bit**

> 解決方法：請重新安裝 (文件？？？？？)

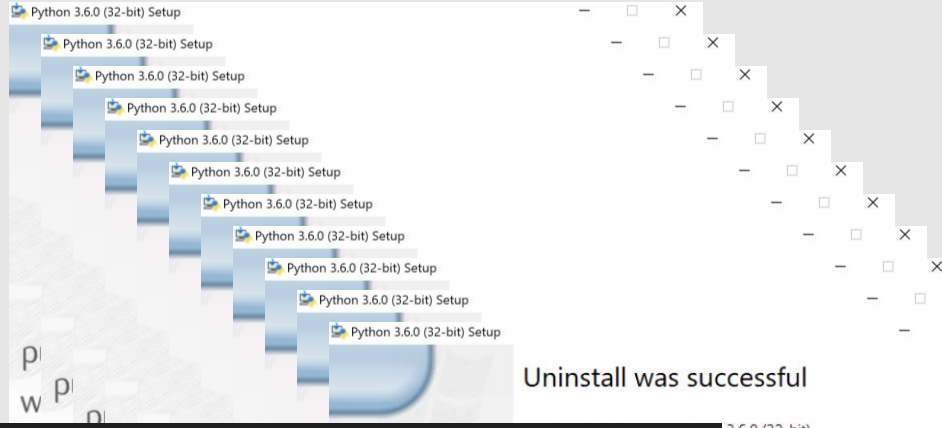
```
[10/1/2019 5:13:36 PM]     from grpc._cython import cygrpc as _cygrpc
[10/1/2019 5:13:36 PM] ImportError: cannot import name 'cygrpc'
[10/1/2019 5:13:36 PM] Stopping host...
[10/1/2019 5:13:36 PM] Stopping JobHost
[10/1/2019 5:13:36 PM] Job host stopped
[10/1/2019 5:13:36 PM] Host shutdown completed.
```



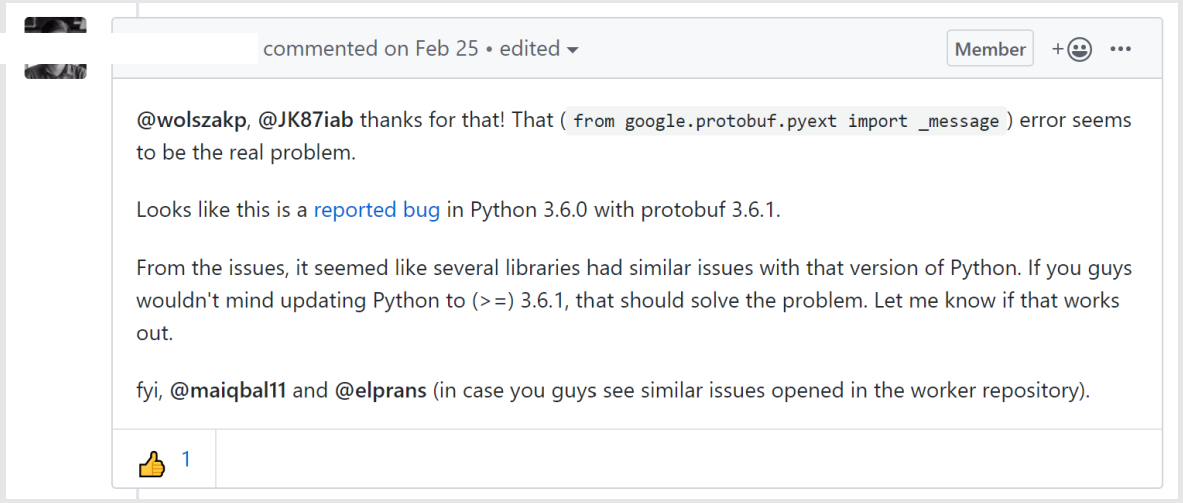
需要注意

- 請安裝 **Python >= 3.6.1** 版本

> 解決方法：請重新安裝 (文件裝飾用？？)




```
[10/1/2019 5:59:28 PM]   File "C:\Users\huihuan\AppData\Roaming\npm\node_modules\azure-functions-core-tools\bin\workers\python\deps\azure_
functions_worker\main.py", line 7, in <module>
[10/1/2019 5:59:28 PM]     from . import dispatcher[10/1/2019 5:59:28 PM] Exceeded language worker restart retry count for runtime:python.
 Shutting down Functions Host

[10/1/2019 5:59:28 PM]   File "C:\Users\huihuan\AppData\Roaming\npm\node_modules\azure-functions-core-tools\bin\workers\python\deps\azure_
functions_worker\dispatcher.py", line 19, in <module>
[10/1/2019 5:59:28 PM]     from . import bindings
[10/1/2019 5:59:28 PM]   File "C:\Users\huihuan\AppData\Roaming\npm\node_modules\azure-functions-core-tools\bin\workers\python\deps\azure_
functions_worker\bindings\__init__.py", line 2, in <module>
[10/1/2019 5:59:28 PM]     from .meta import check_input_type_annotation
```



[redacted] commented on Feb 25 • edited Member

@wolszakp, @JK87iab thanks for that! That (`from google.protobuf.pyext import _message`) error seems to be the real problem.

Looks like this is a reported bug in Python 3.6.0 with protobuf 3.6.1.

From the issues, it seemed like several libraries had similar issues with that version of Python. If you guys wouldn't mind updating Python to (>=) 3.6.1, that should solve the problem. Let me know if that works out.

fyi, @maiqbal11 and @elprans (in case you guys see similar issues opened in the worker repository).

1

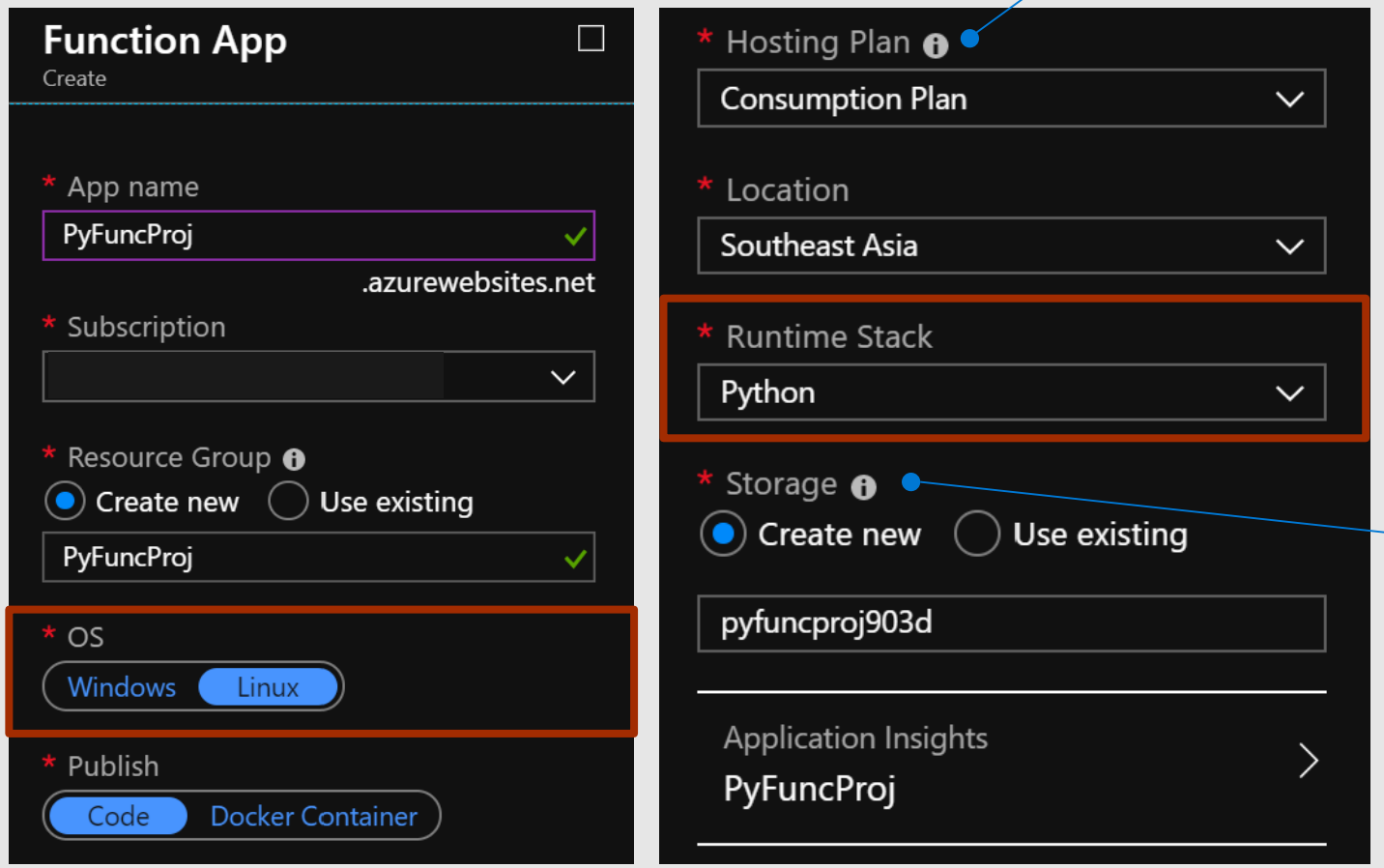
記得！記得！要把虛擬環境跑起來並安裝 requirements.txt 的相關套件
撰寫完畢後也要把相對應的套件寫道 requirements.txt

部屬 – 建立資源

- 使用 Azure Portal
- 使用 Visual Studio Code
- 使用 Visual Studio
- 使用 Azure CLI

部屬 – 建立資源

- 使用 Azure Portal



Hosting Plan (主控服務)
> 使用 Consumption Plan

Consumption Plan (取用方案)

- 只有函式執行時才需付費
- 自動調整規模，即使在高負載期間也會
- 函式執行會在一段可設定的時間之後逾時

App Service Plan (專用方案)

- 提供預約的伺服器
- 費用與其他 App Service 資源相同
- 藉由新增更多 instance 來手動增大或啟用自動調整
- 較容易預測每月成本

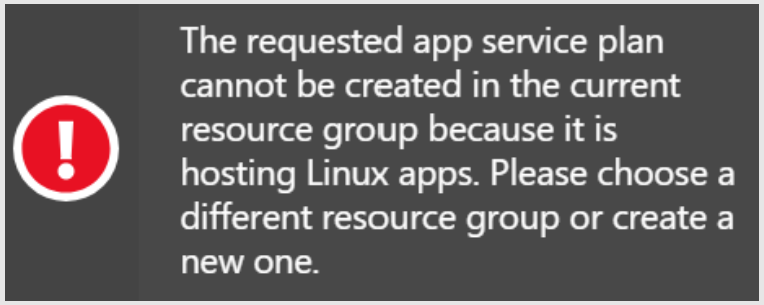
Storage
> 使用同一個 Storage Account

- 建立函數應用程式時須建立或連結至支援 Blob、佇列和資料表儲存體的 Azure 儲存體
- Azure Functions 會在內部使用 Azure 儲存體來進行作業，如管理觸發程序和記錄函式執行
- 使用 Consumption Plan 時，函式程式碼和繫結組態檔會儲存在 Azure 儲存體中

部屬 – 需要注意

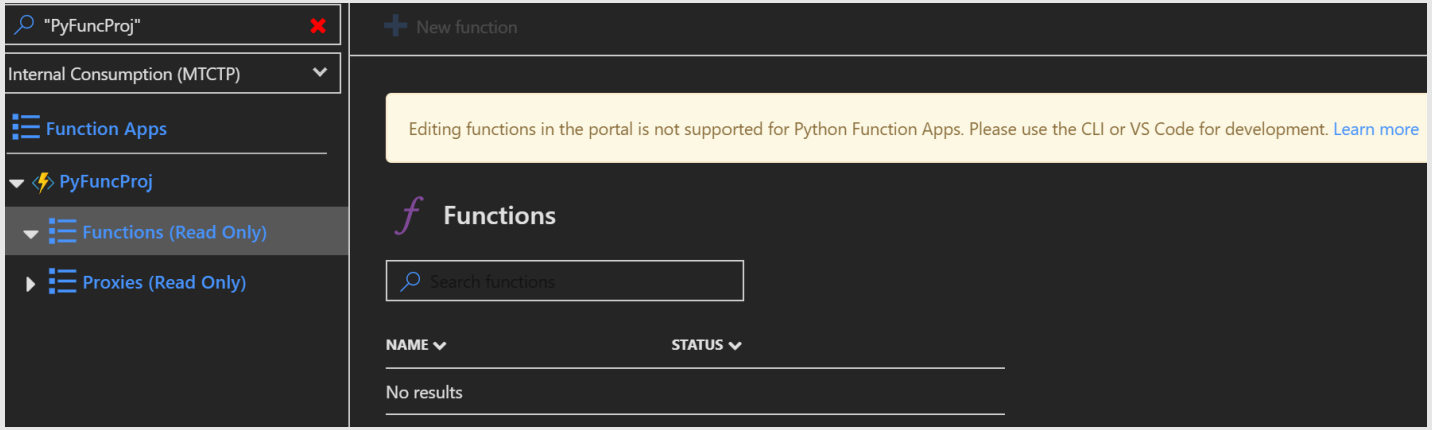
- 在**同資源群組**內開**同區域**的 **Window 及 Linux** 的函式，是**不可以**的！

> 解決方法：開不同的區域或分不同資源群組



- 在 **Linux** 的函式是**不支援**在 Azure Portal 上新增 / 修改函式內容

> 解決方法：使用 Azure CLI 或 VS Code 來做部屬及修改



部屬 – 手動部屬

- 使用 Azure Core Tool 執行 `func azure functionapp publish <app name> **--build remote**` 將專案內的函式都部屬到雲端 Azure Functions 的環境
  - 須將所有相依性都列在 requirements.txt 檔案中，且檔案位於專案目錄的根目錄中
  - 透過 `--build remote`，Azure Functions 從遠端建立這些相依性

```
Remote build succeeded!
Syncing triggers...
Functions in PyFuncProj:
    PyEventGridTrigger - [eventGridTrigger]

    PyHttpTrigger - [httpTrigger]
        Invoke url: https://pyfuncproj.azurewebsites                                                                    =

    PyTimerTrigger - [timerTrigger]
```

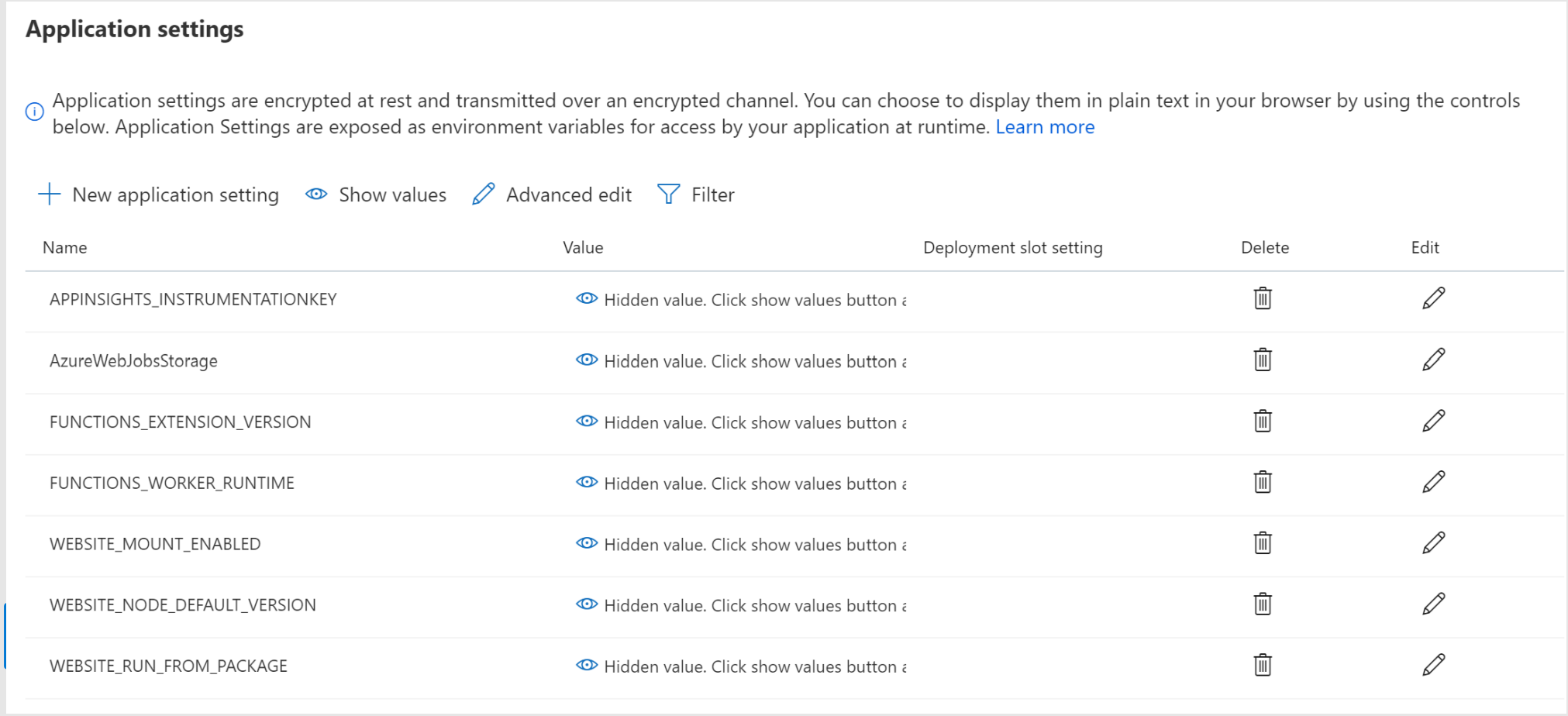
部屬 – 手動部屬

- 使用 Azure Core Tool 執行 `func azure functionapp publish <app name> --publish-settings-only -o` 將專案內 local.settings.json 的設定部屬到 Azure Functions

```
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 39.15 ms for C:\Users\huihuan\source\repos\PythonFuncProj\extensions.csproj.
  extensions -> C:\Users\huihuan\source\repos\PythonFuncProj\bin\extensions.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:01.59


Setting FUNCTIONS_WORKER_RUNTIME = ****
Setting AzureWebJobsStorage = ****
Setting MyCustomLocalSettings = ****
```

部屬 – 手動部屬

- 或直接到 Azure Portal 的 configuration 設定 Application Settings



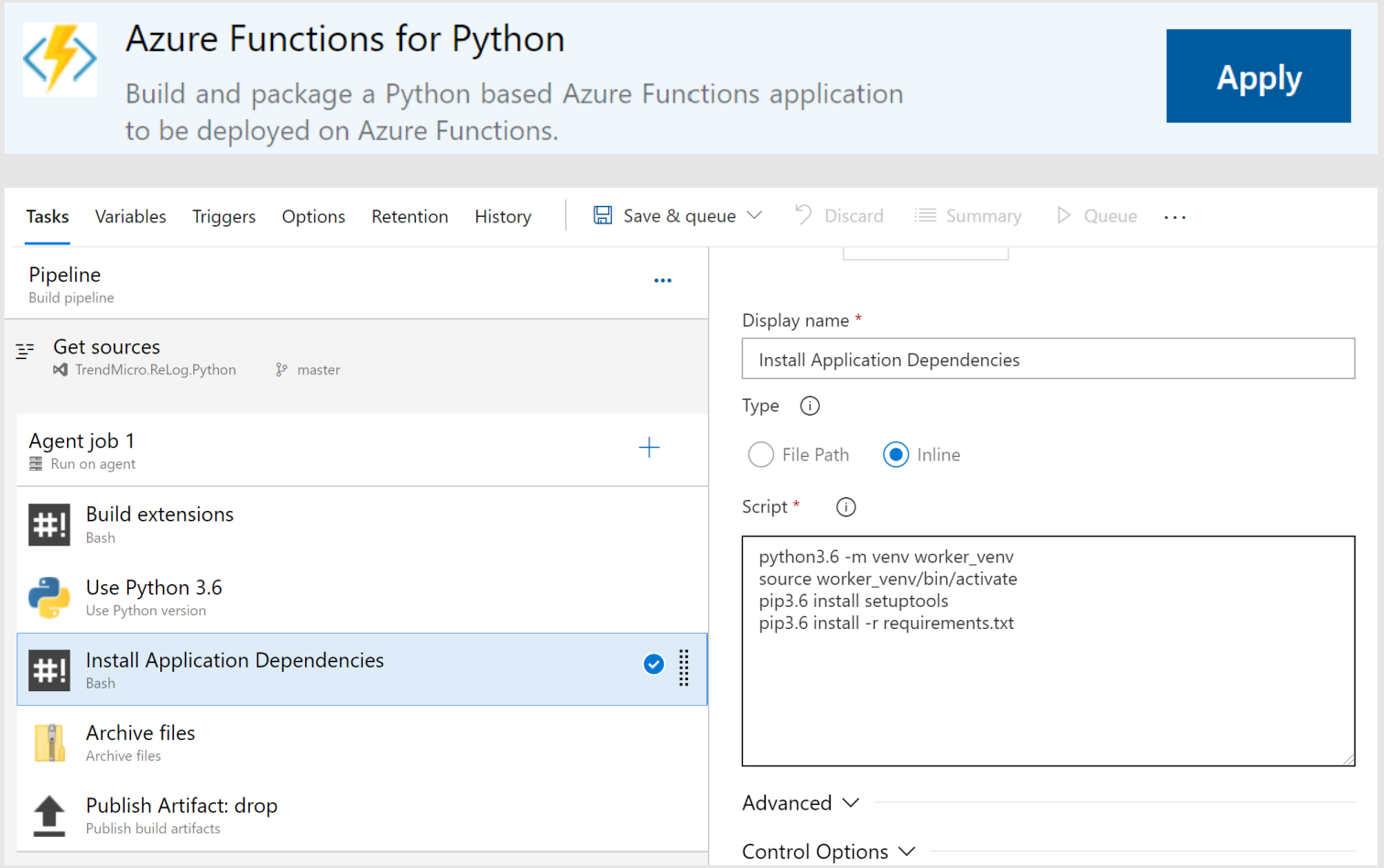
部屬 – 手動部屬

- 若專案中有使用到的dependency在build remote時執行失敗，可以使用docker image的方式做部屬，執行 func azure functionapp publish <app name> **--build-native-deps**
  - 本機需安裝 docker
  - 基本上，Core Tools 會拉取 mcr.microsoft.com/azure-functions/python 映像作為本機上的容器，再使用此環境建立並安裝必要的模組，最後再將其封裝部署至 Azure Functions

```
Getting site publishing info...
Running 'docker cp 4c68e2:"/.python_packages/." "C:\Users\huihuan\source\repos\PythonFuncProj\.python_packages"'..done
Running 'docker kill 4c68e2'..done
Running 'docker exec -t 764f6f chmod +x /ziptofs.sh'..done
Running 'docker exec -t 764f6f /ziptofs.sh'.....done
Running 'docker cp 764f6f:"/file.squashfs" "C:\Users\huihuan\AppData\Local\Temp\tmp2772.tmp"'..done
Running 'docker kill 764f6f'..done
Uploading package...
Uploading 13.13 MB [##############################################################################]
Upload completed successfully.
Deployment completed successfully.
Syncing triggers...
Functions in PyFuncProj:
    PyEventGridTrigger - [eventGridTrigger]

    PyHttpTrigger - [httpTrigger]
        Invoke url: https://pyfuncproj

    PyTimerTrigger - [timerTrigger]
```

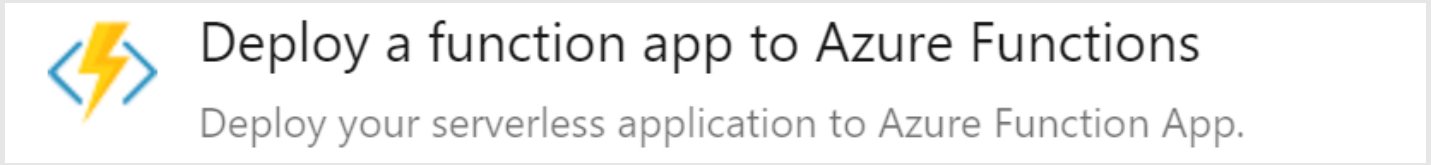
部屬 – CI/CD

- 使用超級好用的 Azure DevOps Service 的 pipelines 來實作 CI/CD
  - Builds：使用 Azure Functions for Python 的 Template



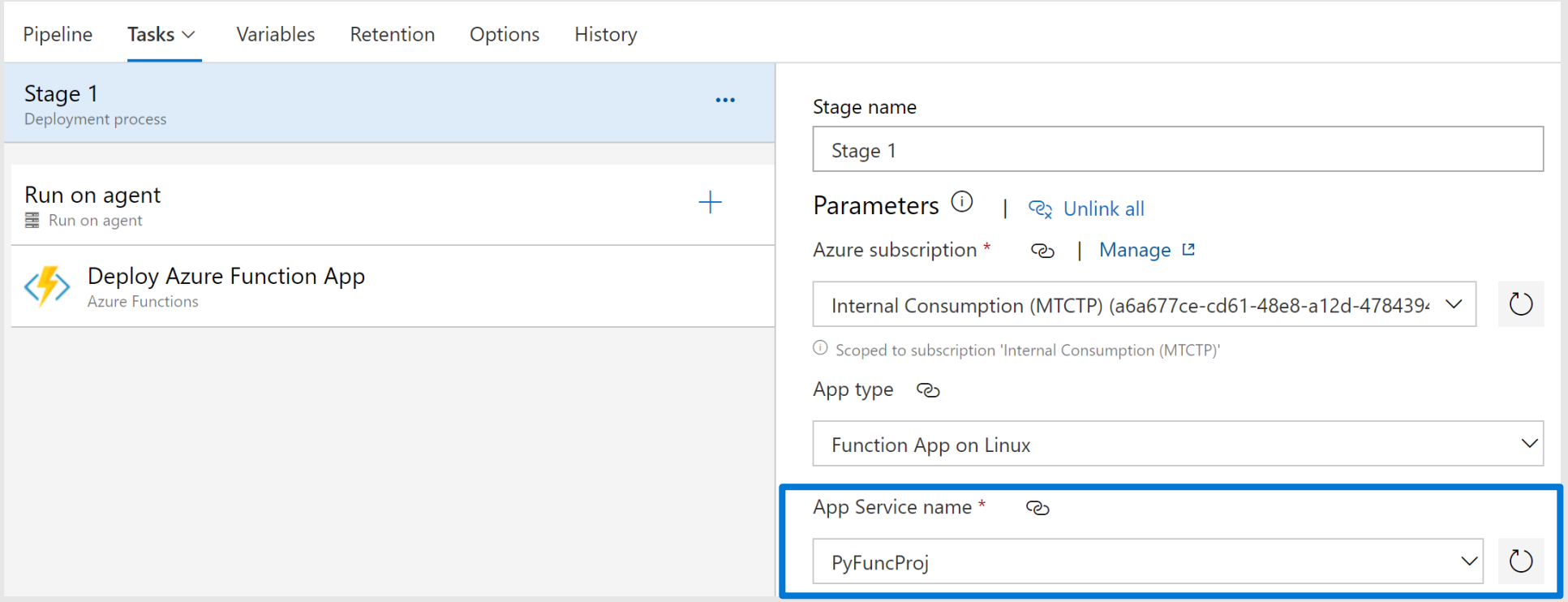
部屬 – CI/CD

- Releases：使用 Deploy a Function App to Azure Functions 的 Template



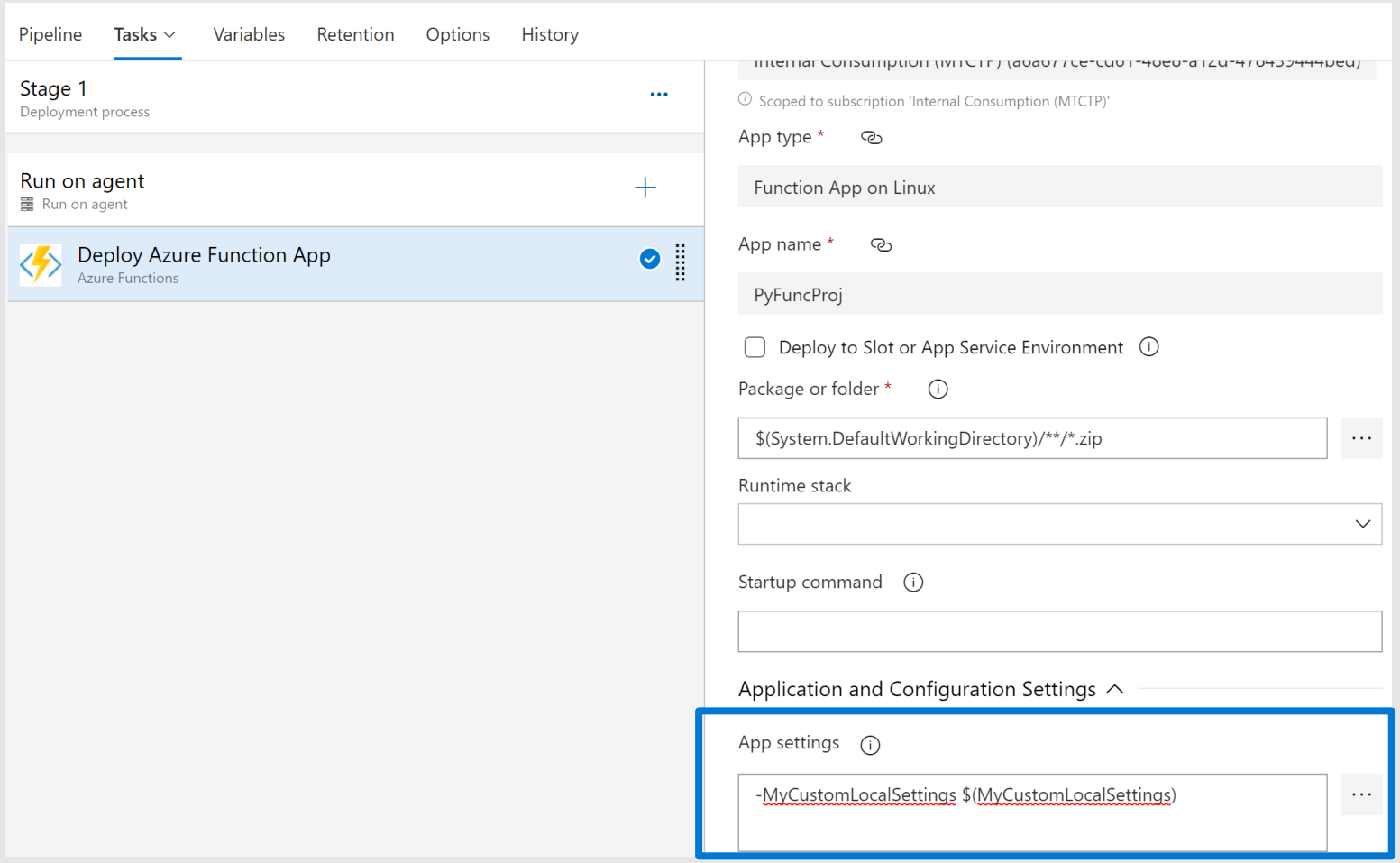
部屬 – CI/CD

- App Service Name 選擇先前在 Azure 上建立的資源



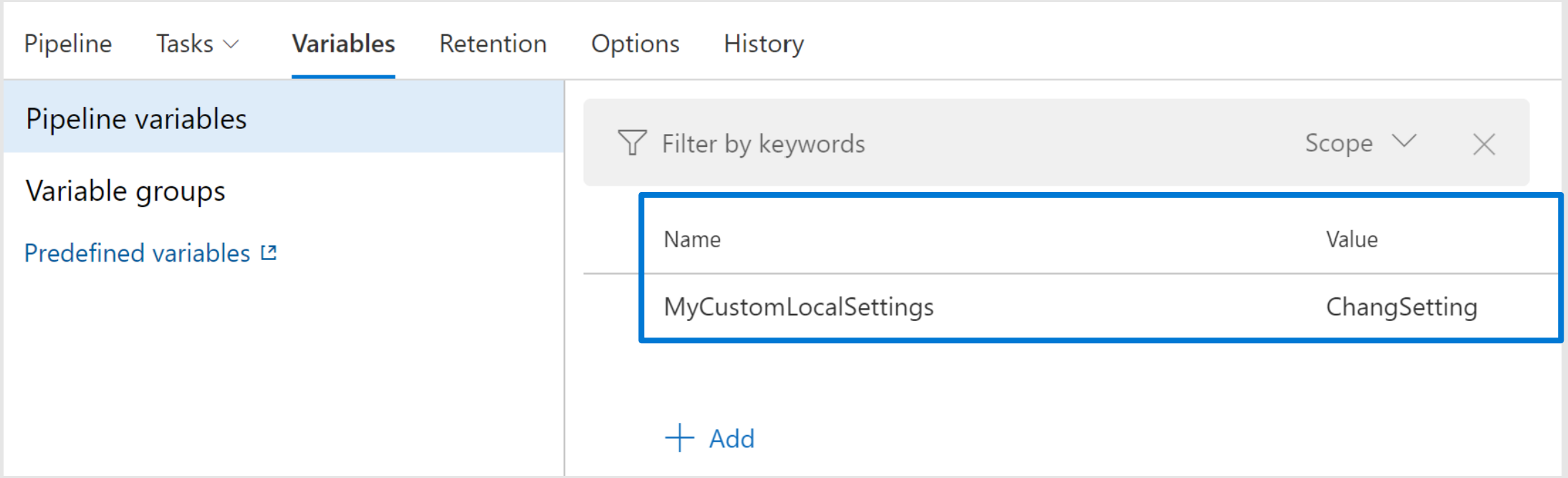
部屬 – CI/CD

- 在 Deploy Azure Function App 的 APP Settings 可以在這裡設定部屬的同時將連線字串等資料部屬到Azure Functions的Application Settings中



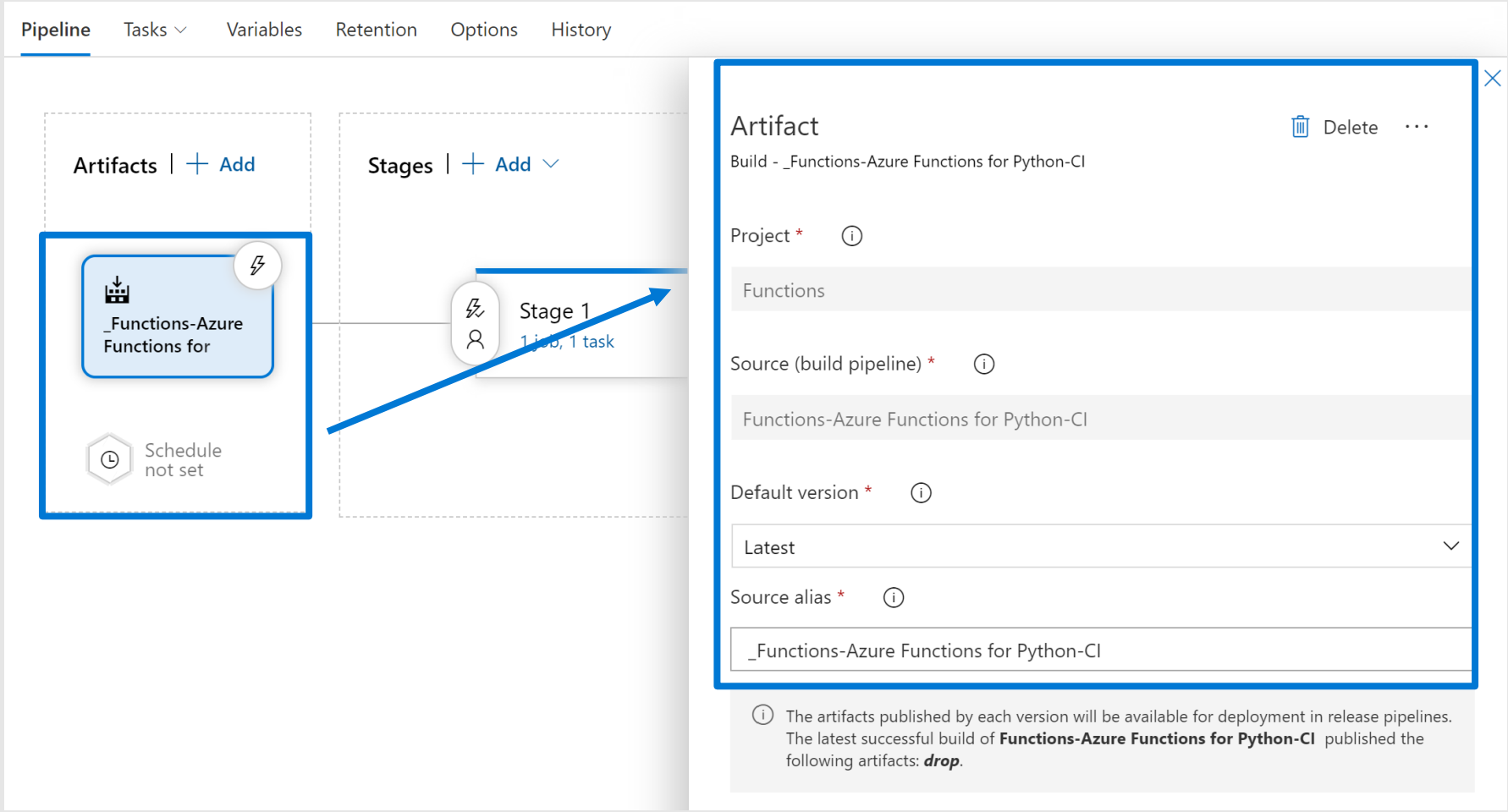
部屬 – CI/CD

- 將連線字串放在 Variables 中



部屬 – CI/CD

- 最後設定要部屬的檔案從哪條 build pipeline 來及要部屬的版號



設定完成後，往後只需在本機開發完，將程式碼 push 到 repo，
Azure DevOps 就會自動幫你部屬啦！

Real Use Case

User Story

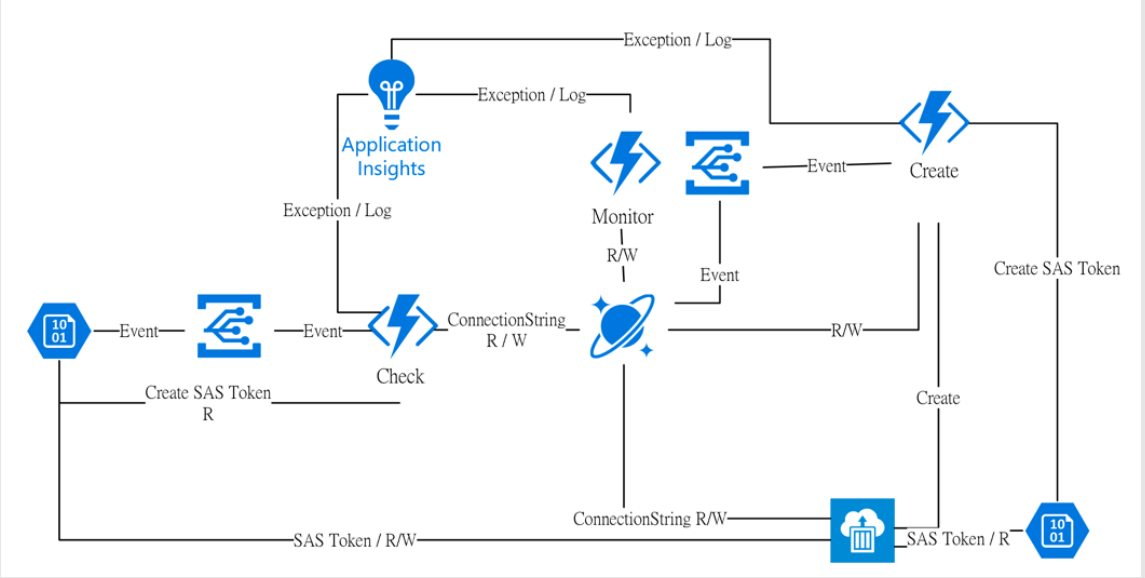
- 身為一個網站管理人員，我希望可以將上傳的 jpg，進行縮圖成 jpeg，以利我能利用小圖來顯示，減省頻寬

- 身為一個使用者，我希望可以將我的影片進行轉檔，以利我觀看影片的時候，可以有不同的解析度

該怎麼設計這個軟體？

- 上傳上去的時候，順便轉檔，並儲存
- 寫一個 Schedule Job，定期去撈取 Storage File 並轉檔
- 還有很多方法

Flow



- Storage 有檔案進來或更新
- 由 Storage 更新而觸發 CheckMD5Func 比對檔案
  - 透過 MD5 進行比對是否有更動
  - 產生 SAS Token
  - 在 Cosmos 新增一筆新資料或更新原有資料
- 由 Cosmos 更新而觸發 CreateACIFunc 建立 ACI
  - 產生 SAS Token
  - 因無法預測轉檔執行的時間，故僅拋出 Windows ACI 建立的 request
  - 更新 Cosmos 資料
- ACI 撈取放在 Storage 的 PowerShell Script 執行
  - Powershell Script 負責處理轉檔，轉檔完成後檔案寫回原 Blob
  - 更新 Cosmos 資料
- 透過 MonitorFunc 監控來確保整個流程
  - 若ACI執行完成則將ACI刪除
  - 30 分鐘內未完成，透過更新 Cosmos資料來啟動重新轉檔流程
  - 失敗超過指定的次數則停止建立
  - 可以透過調整容忍失敗次數來重新啟動轉檔流程

Monitor 監控機制


```
"id": "5073254a-a622-4a5b-94c3-6d55f90d4b94",
"accountName":
"containerName": "logsource",
"resourceGroupName": "huier-logsource",
"fileName": "1/1101 - Copy (10) - Copy - Copy - Copy - Copy - Copy - Copy.blg",
"md5": "4K6Fu2bw1O++zkhrAjkWYg==",
"blobUrl":  orage.blob.core.windows.net/logsource/1/1101 - Copy (10) - Copy
"accessUrl": "?sv=2018-03-28&sig=nF%2BLjqXXL0wKIhzCO%2Bq2ECC3moipoOm5kqbk63BiNwo%3D&spr=https&st
"createDateTime": "2019-07-04T04:20:47.1403488Z",
"updateDateTime": "2019-07-04T04:28:12.176822+00:00",
"relogInfo": {
    "relogStatusType": 7,
    "checkMd5Same": true,
    "isCheckMd5DateTime": "2019-07-04T04:20:47.1403533Z",
    "isCreateAci": true,
    "createAciDateTime": "2019-07-04T04:23:14.537298+00:00",
    "isRelog": true,
    "relogDateTime": "2019-07-04T04:27:20.6723412+00:00",
    "isDone": true,
    "doneDateTime": "2019-07-04T04:28:12.176822+00:00",
    "isRetry": false,
    "retryDateTime": null,
    "reCount": 0,
    "isStop": false,
    "ReWrite": 0,
    "isLogFileExist": true
},
"_rid": "YidIAN+bFCABAAAAAAAAAA==",
"_self": "dbs/YidIAA==/colls/YidIAN+bFCA=/docs/YidIAN+bFCABAAAAAAAAAA==/",
"_etag": "\"00000103-0000-1900-0000-5d1d805c0000\"",
"_attachments": "attachments/",
"_ts": 1562214492
```


- 透過每一動都會將目前的動作記錄到Cosmos裡面，Monitor每一分鐘就去確認一次狀態，以確保整個流程沒有crash

- Cosmos Feed 用來記錄這筆紀錄是否有異動的資料

Issue – ACI 拉取速度太慢

問題狀況：

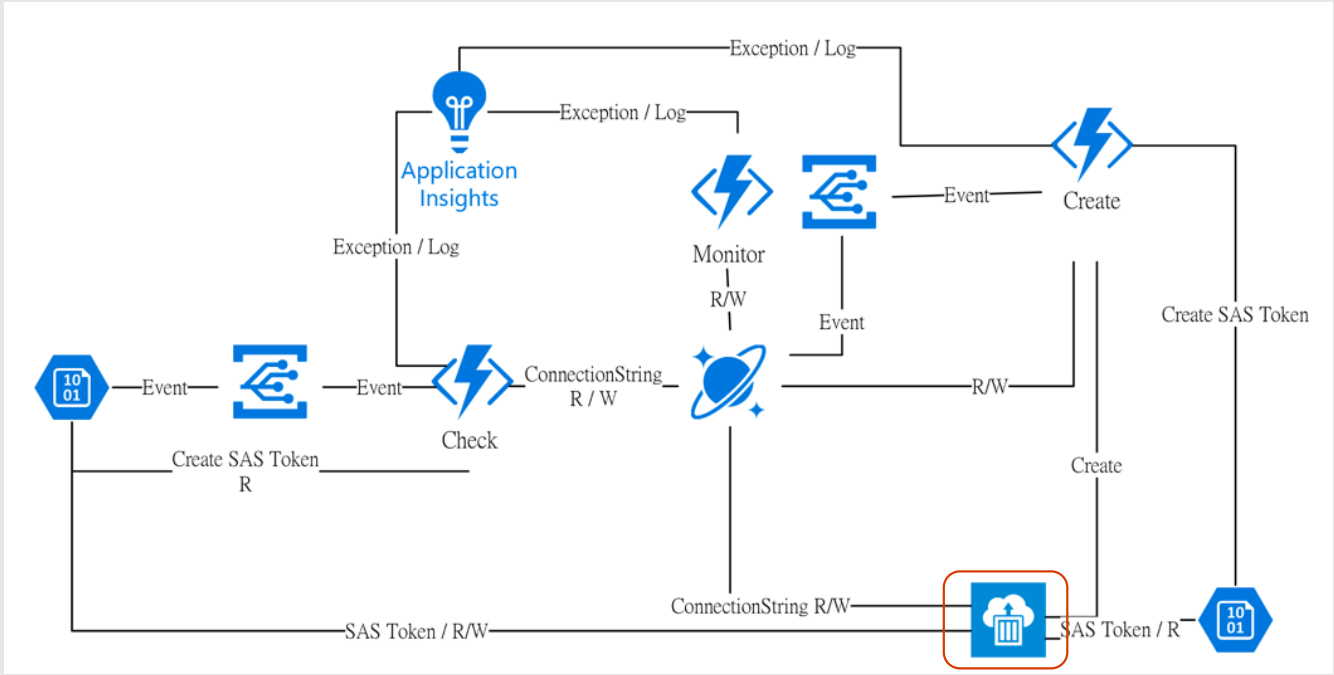
- ACI 建立出來的 Windows Container 拉取太慢

原因：

- Windows Container 太大

解決方案：

- ACR 要使用有 CACHE 的
  (有快取只需要 2min , 沒有的話需要 12 min)



https://docs.microsoft.com/en-us/rest/api/container-instances/listcachedimages/listcachedimages

Issue – ACI 無法連線網路

問題狀況：

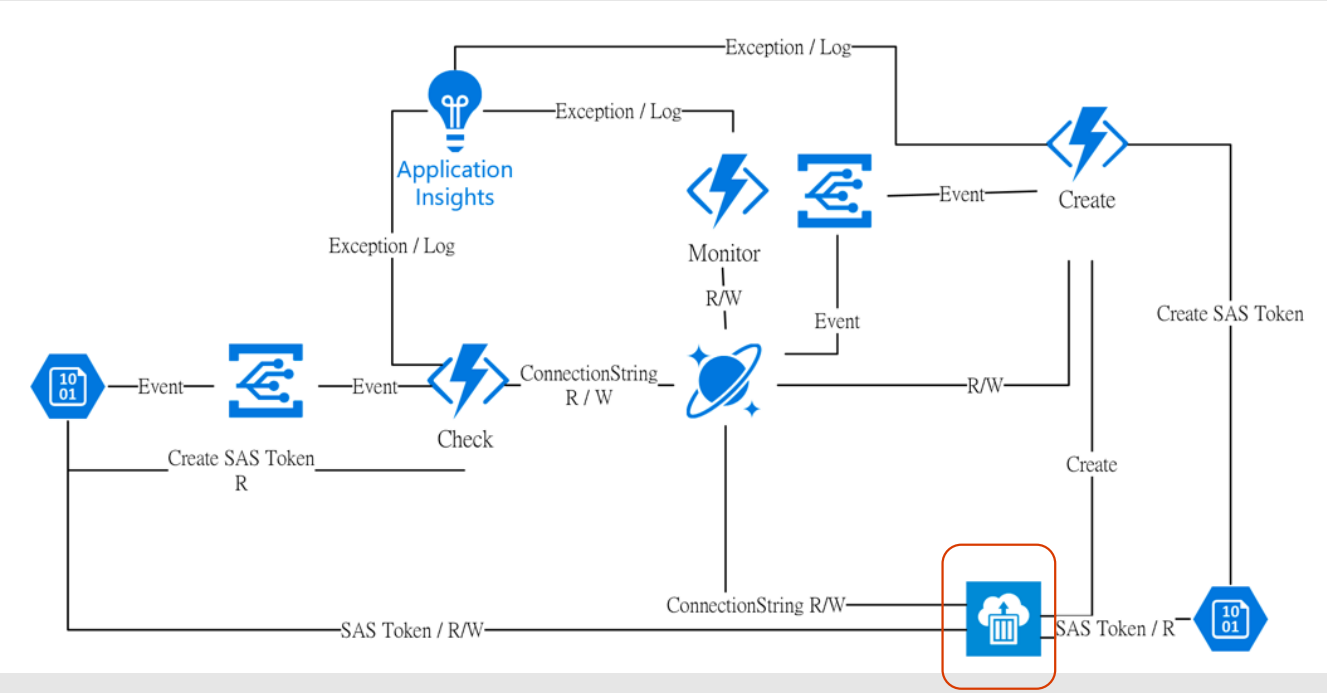
- ACI 建立出來的 Windows Container 無法連線到 Storage

原因：

- 至少需要 30 Sec 的準備時間

解決方案：

- 讓他 Sleep 40 Sec



Issue – ACI 建立數量限制

問題狀況：

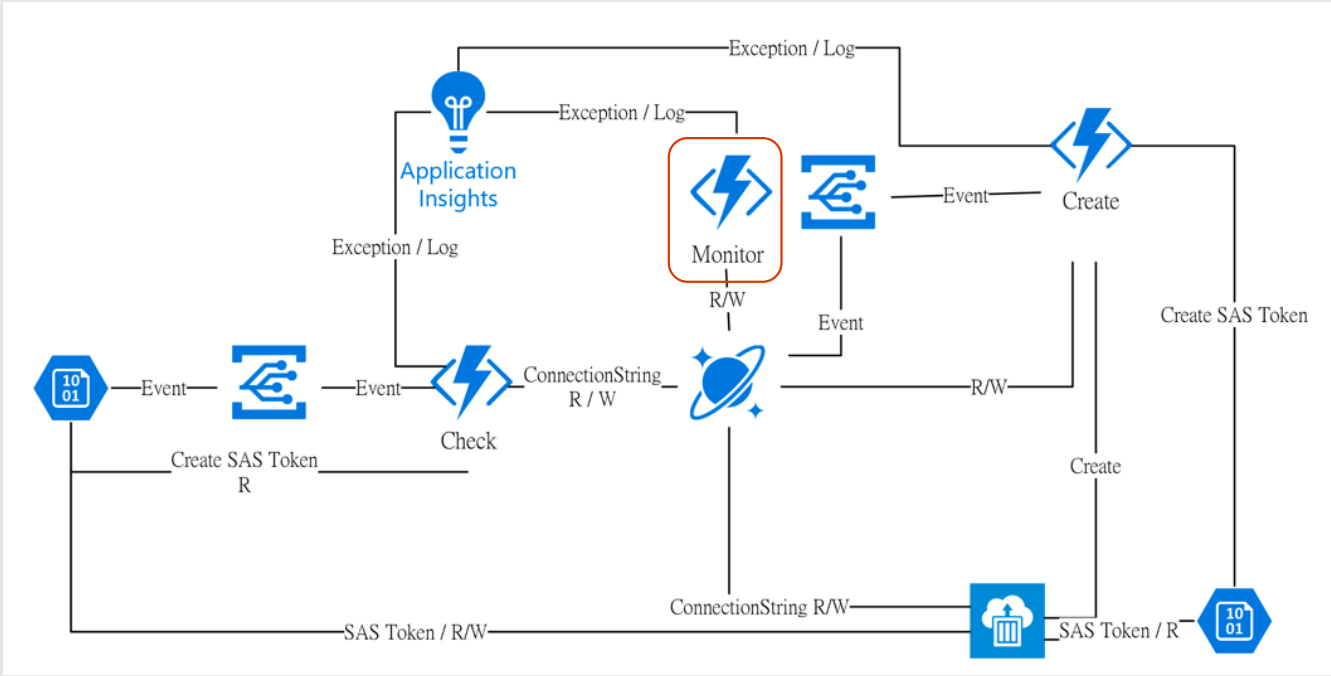
- ACI 開啟失敗

原因：

- ACI 有建立數量的限制

解決方案：

- 建立 Monitor 的機制



心得

- 可以非常專注地在開發程式碼上面，並且支援的功能遠超越我原本的想像
- 真的被炸得很痛
- 不一樣的體驗，其實很有趣啦
- 在 preview 階段開發就是祈禱不是我的程式碼有 bug 是 Azure Function 本身的問題
  (結果最後都是自己的 bug 搞自己)
- 練習怎麼用心平氣和的方式在 github 上面 report bug
  (而且要寫英文 / 然後莫名其妙就被 close 了)
- Preview > GA 又是一個新旅程新體驗

Microsoft

謝謝大家 (๑˃̵ᴗ˂̵)و

小小簡單的介紹

希望大家都能有一 些些些些些 的收穫

如有不周 請海涵 ヾ(((;ꐦ▱ꐦ;)))ノ