中原大學 雲端計算平台實務

12/03-作業報告

Create serverless applications

資訊四乙 10727211 林彥輝

授課教師：鍾武君 教授

中華民國一一〇年十二月

1. Learning Path Intro

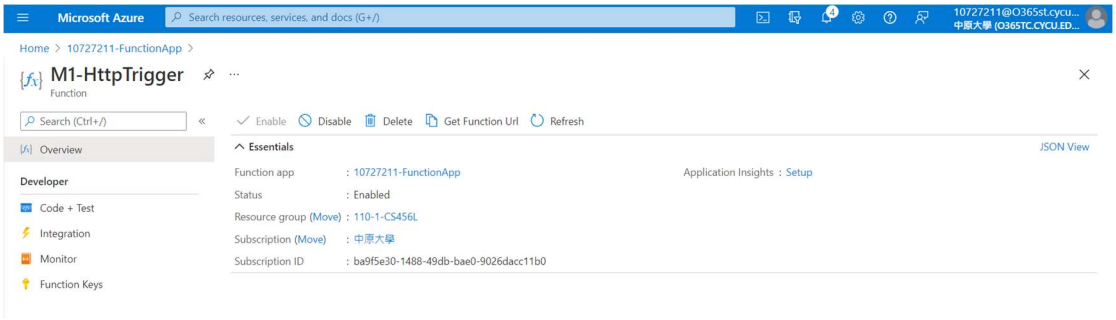
Create serverless applications

https://docs.microsoft.com/en-us/learn/paths/create-serverless-applications/

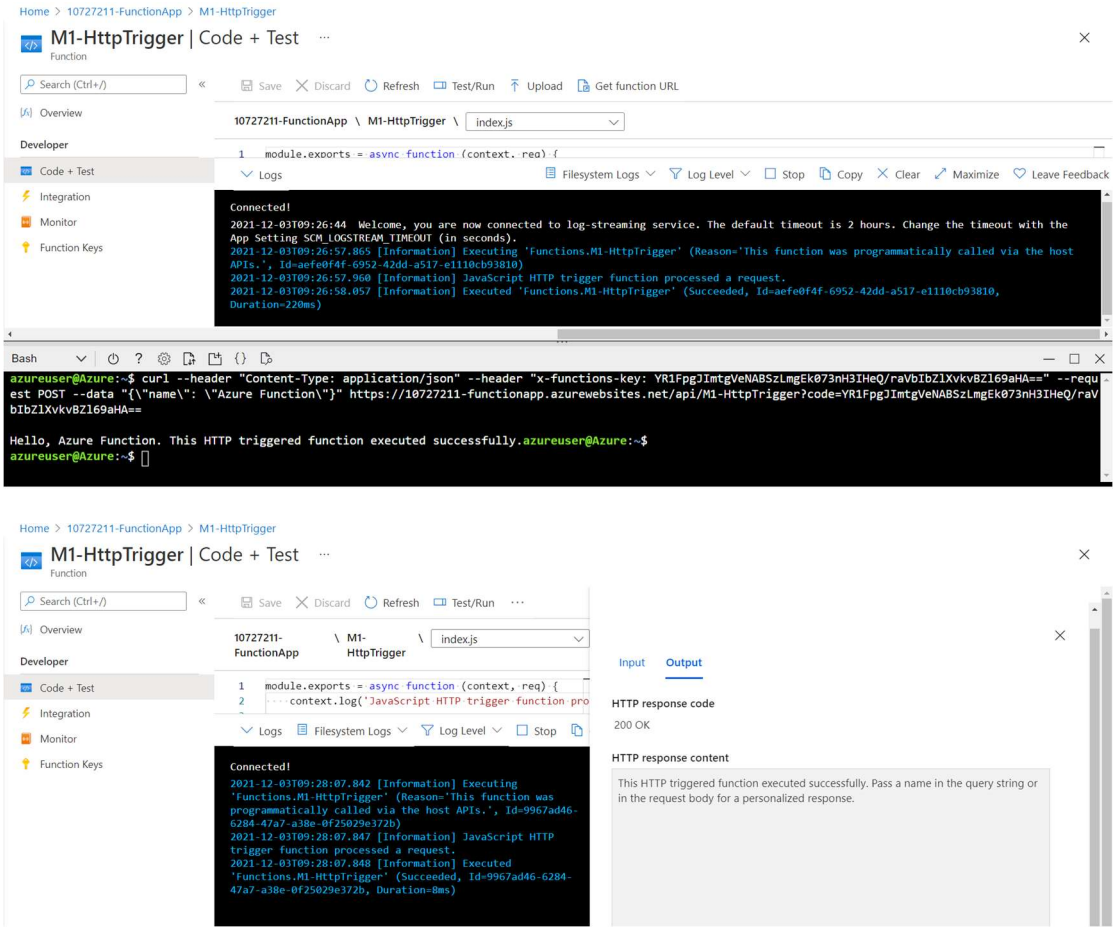
2. Summary Homework Assignment

Model 2:Create serverless logic with Azure Functions

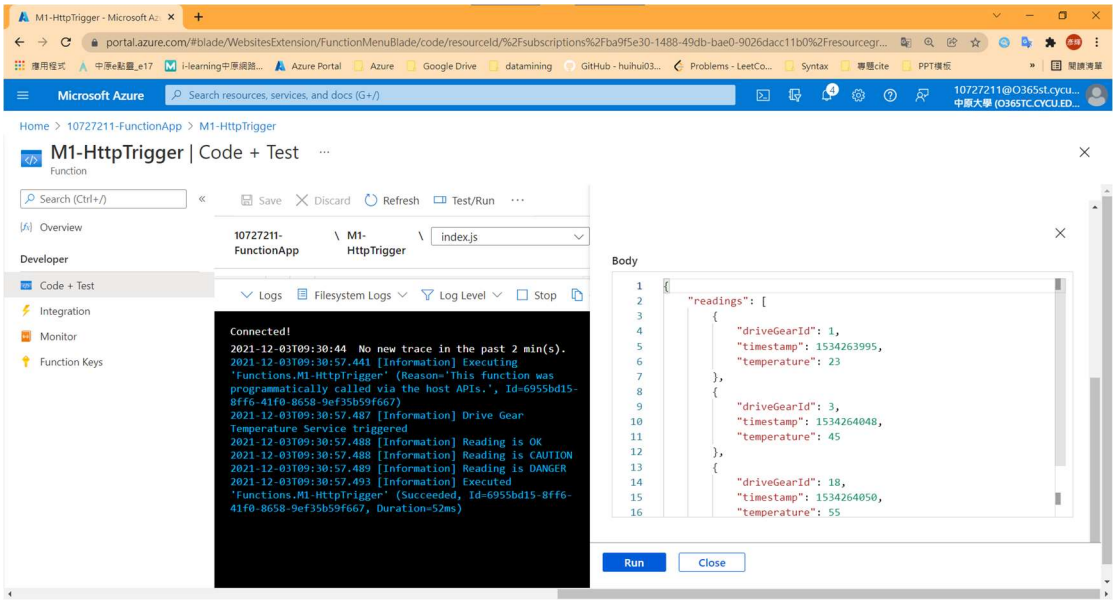
1. Create a function app



2. Test Function - 1

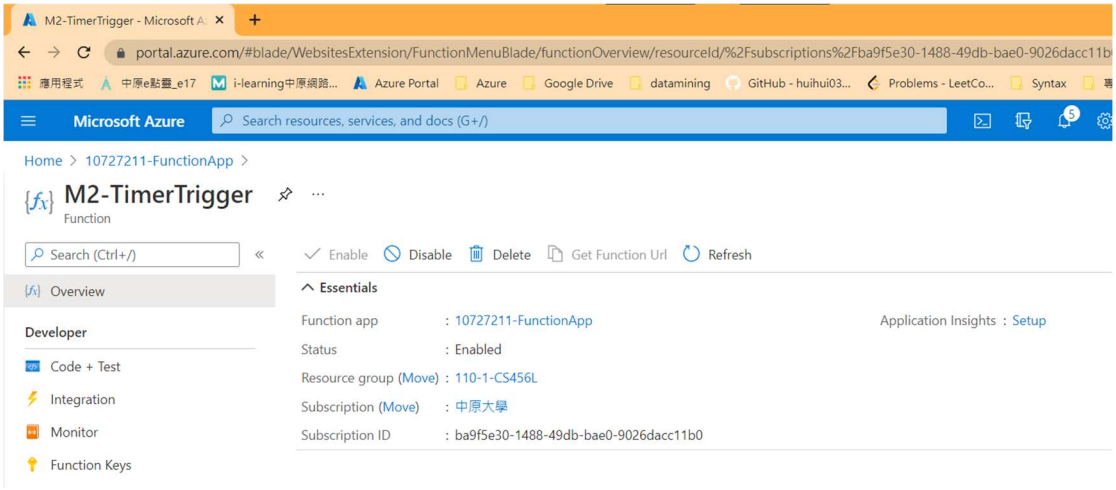


3. Test Function – 2 (IOT temperature)

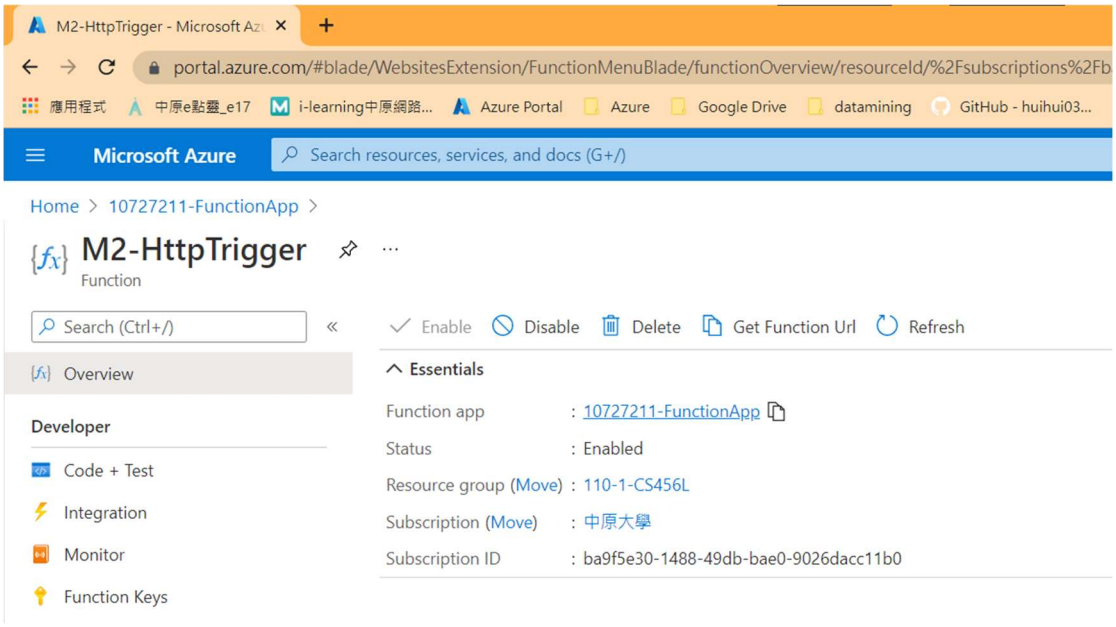


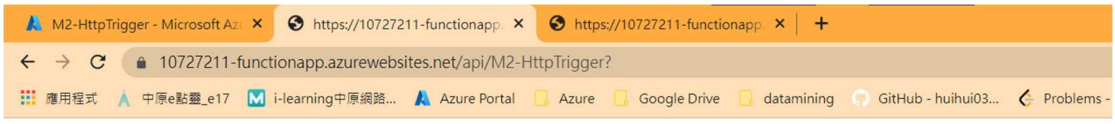
Model 3: Execute an Azure Function with triggers

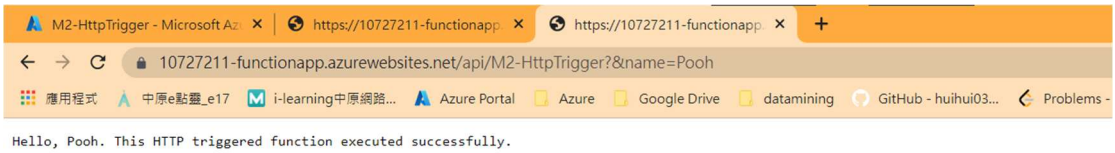
1. Create TimerTrigger & Test



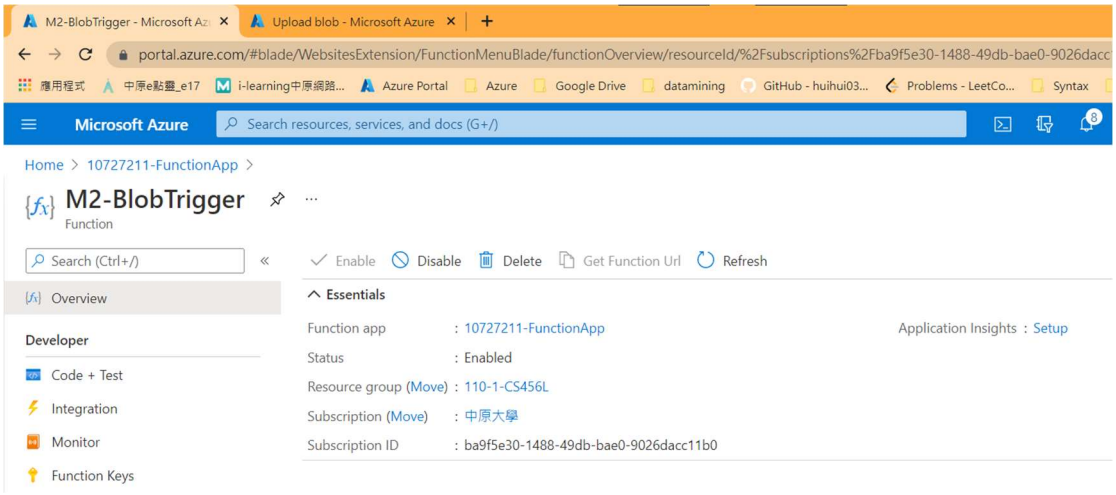
2. Create HttpTrigger & Test

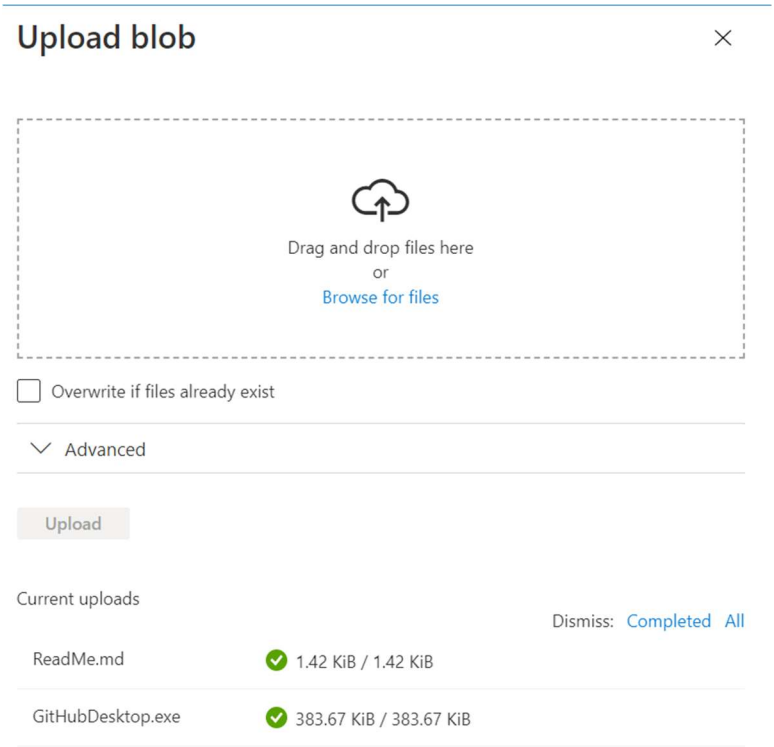


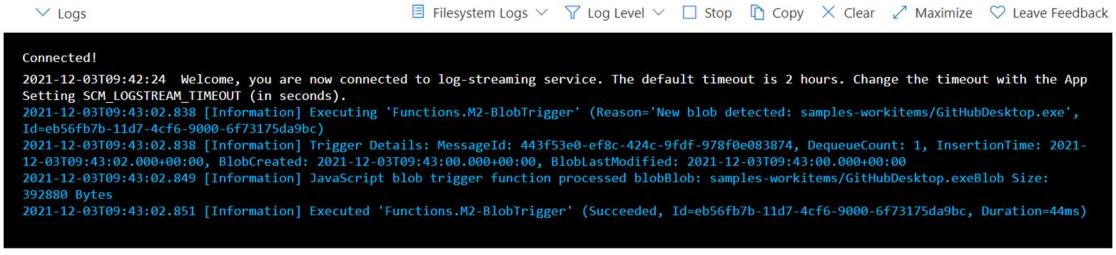




3. Create BlobTrigger & Test

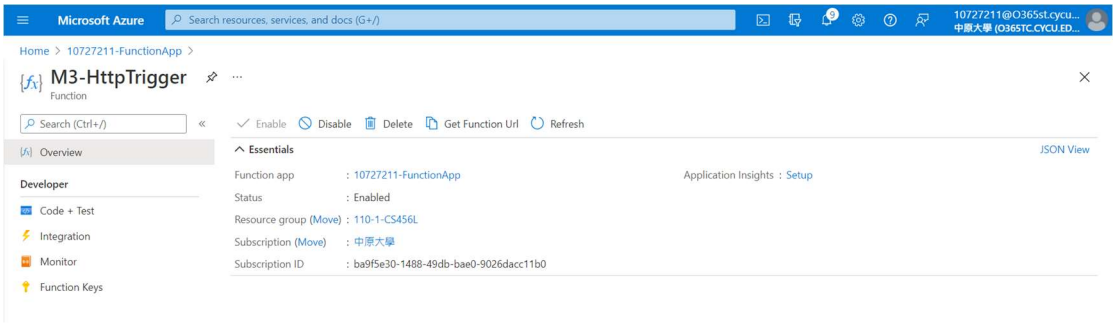




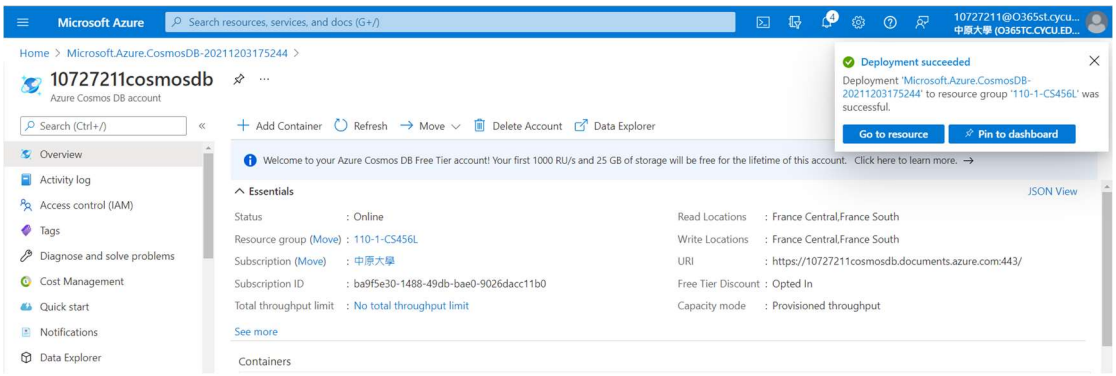


Model 4: Chain Azure Functions together using input and output bindings

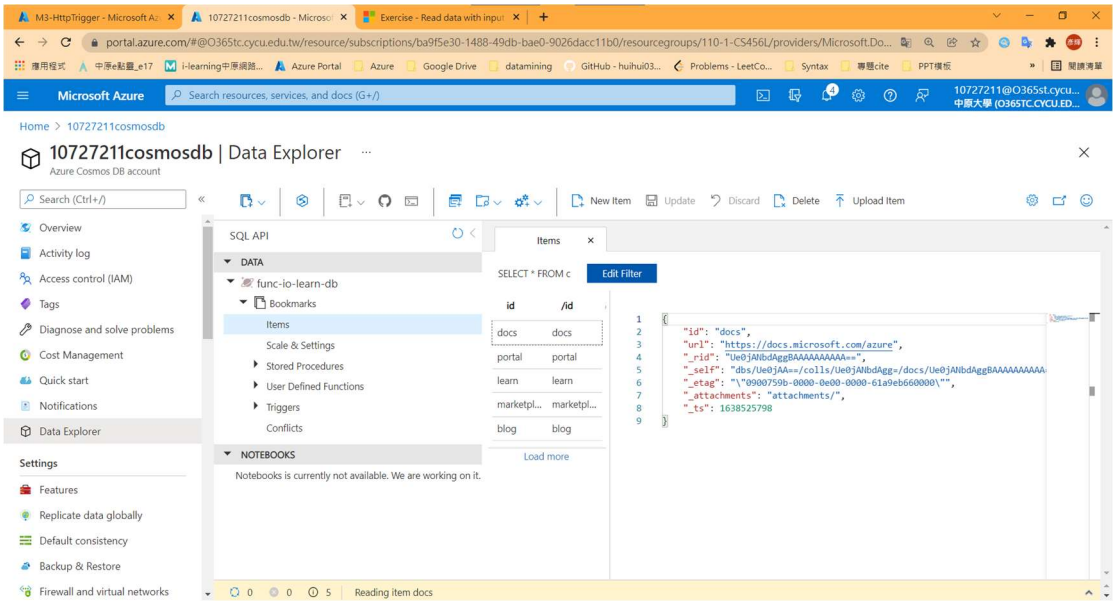
1. Create a HttpTrigger



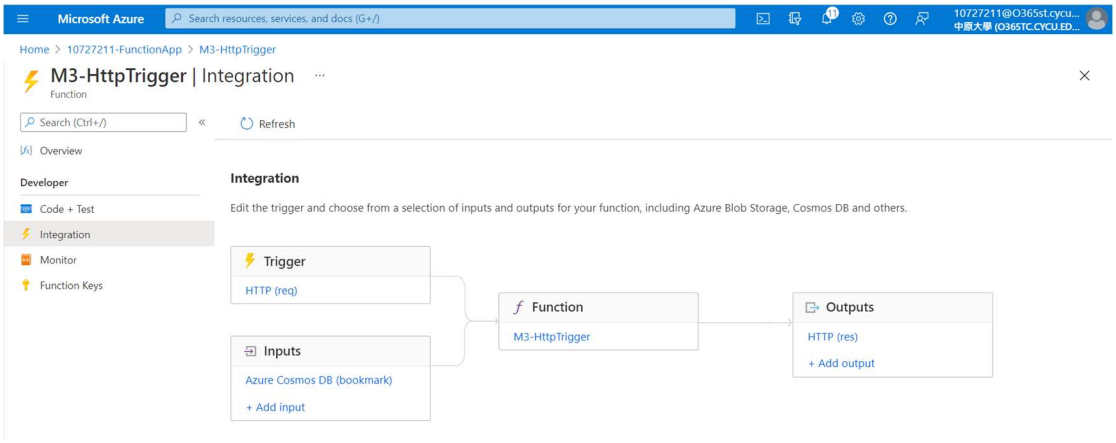
2. Create a cosmosDB



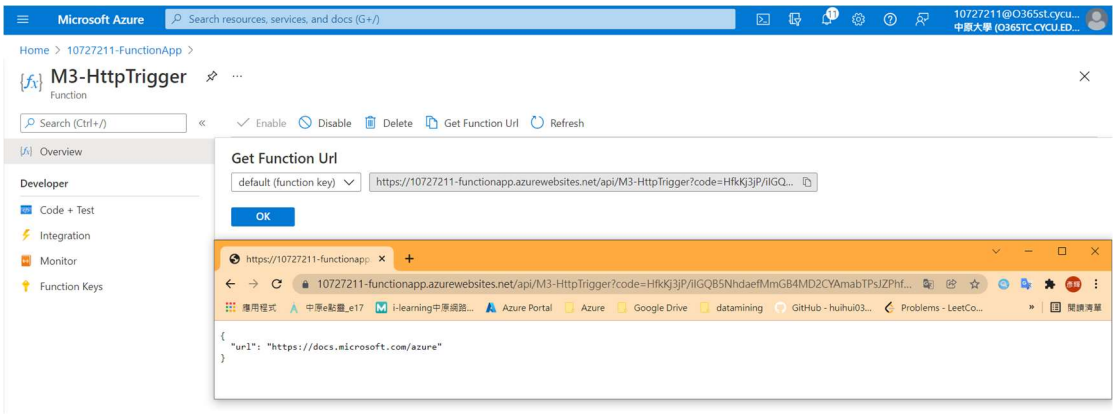
3. Add test data



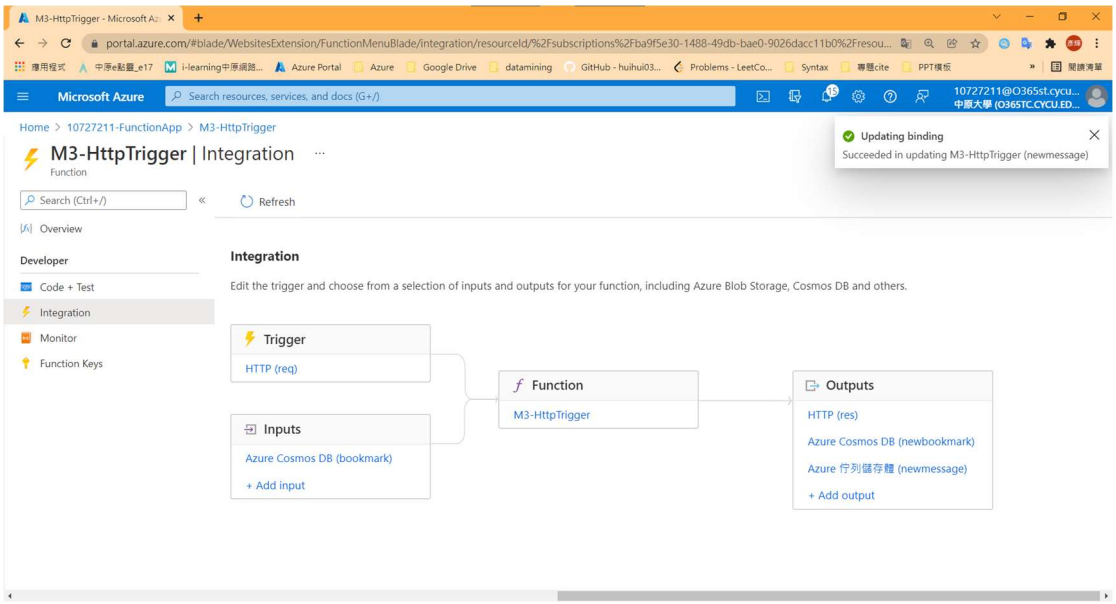
4. Add Azure cosmosDB binding



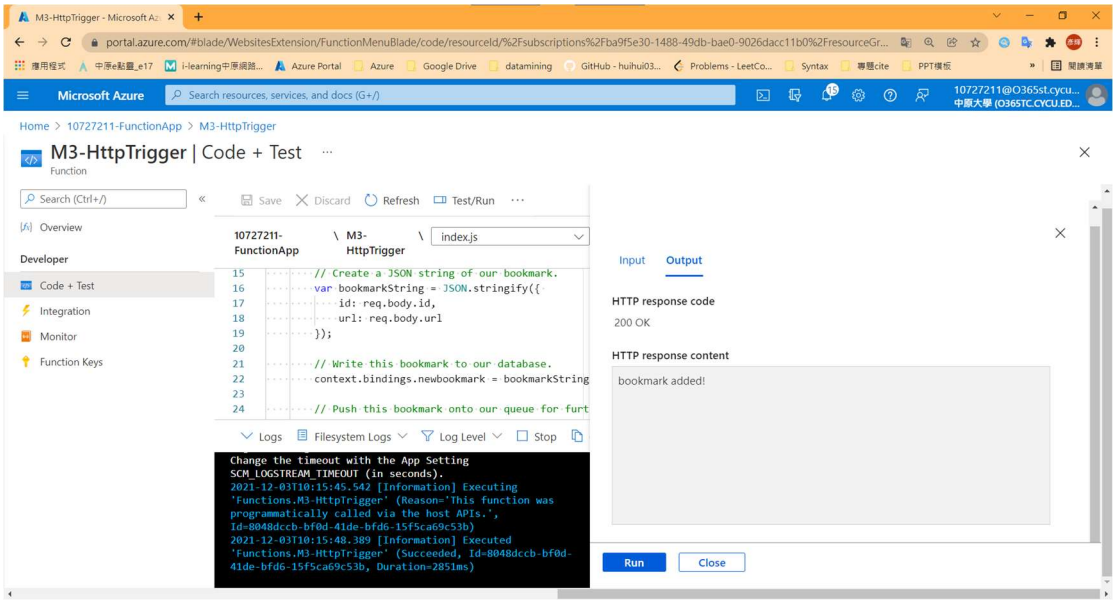
5. Result



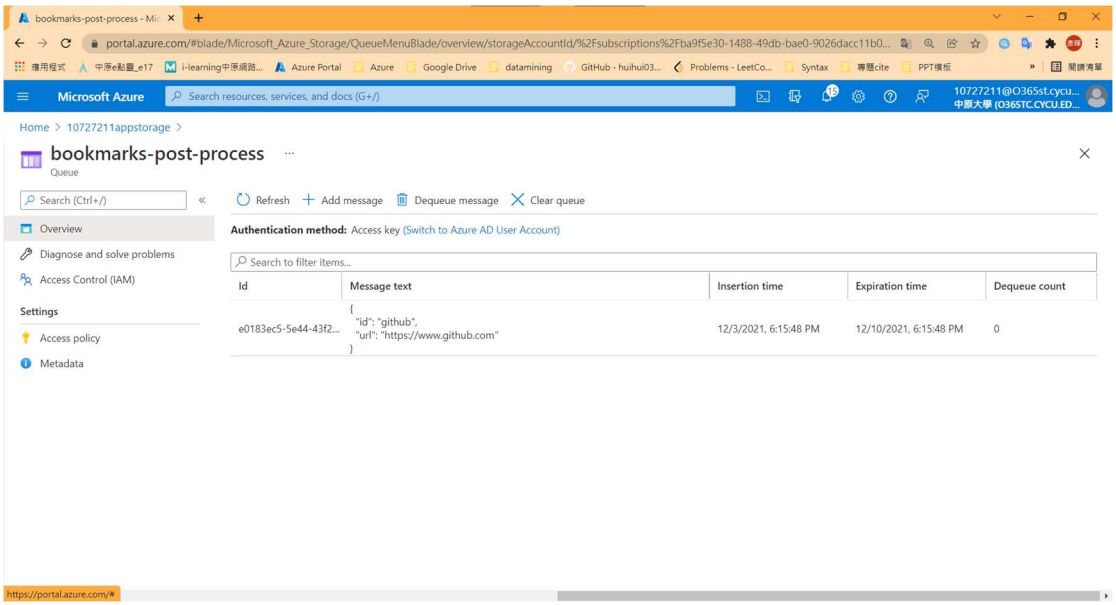
6. Add an Azure Queue Storage and cosmosDB output binding



7. Push message to queue storage

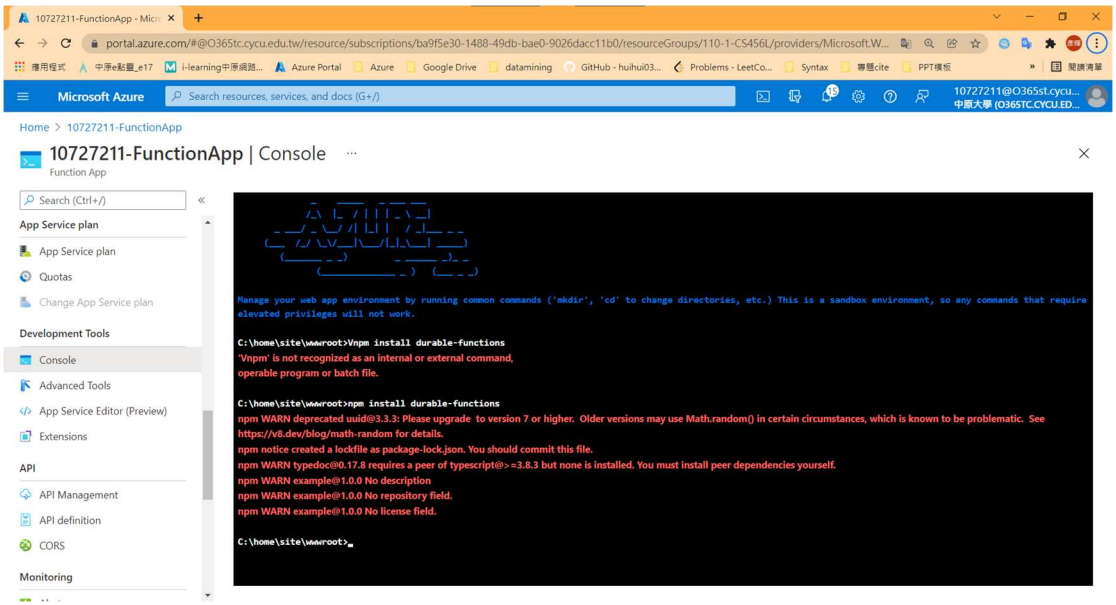


8. Result

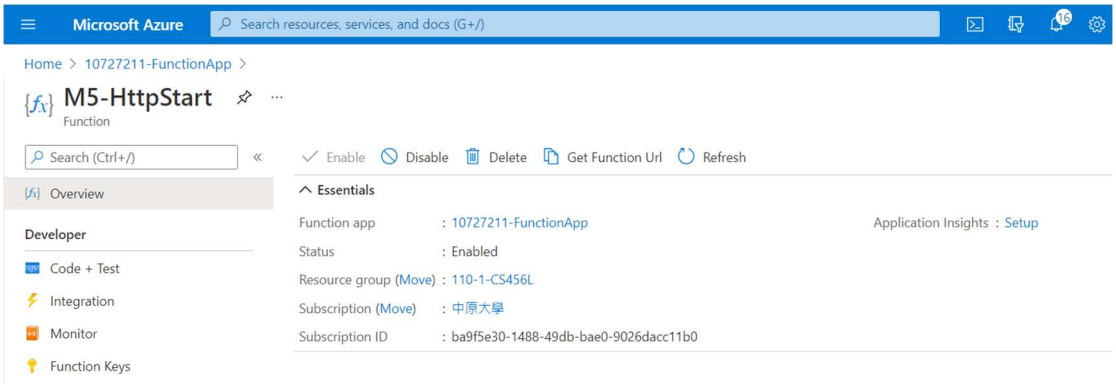


Model 5: Create a long-running serverless workflow with Durable Functions

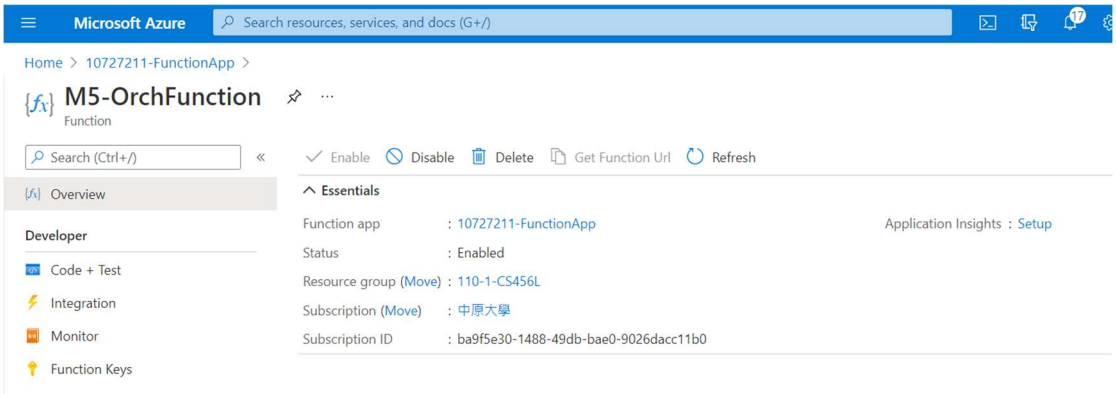
1. Install the durable-functions npm package

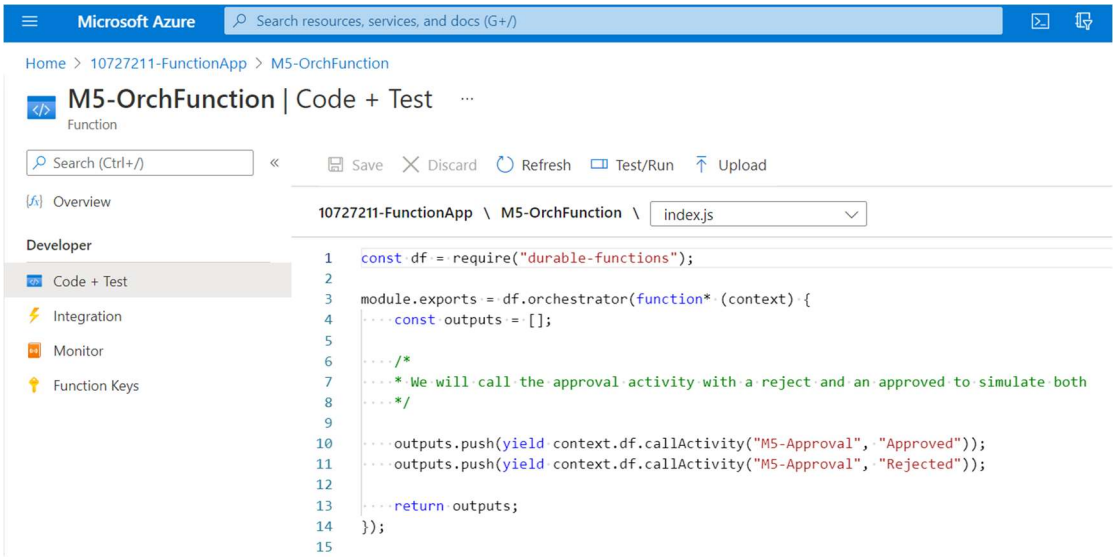


2. Create Durable Functions Http start

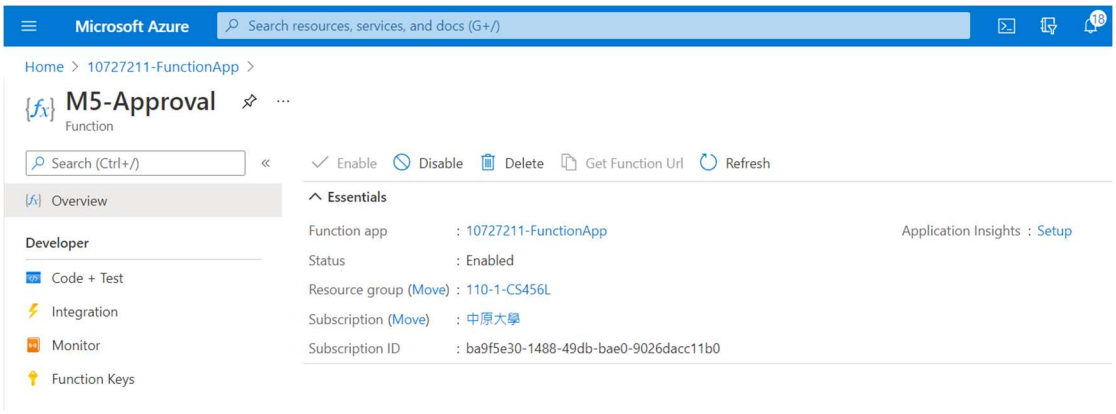


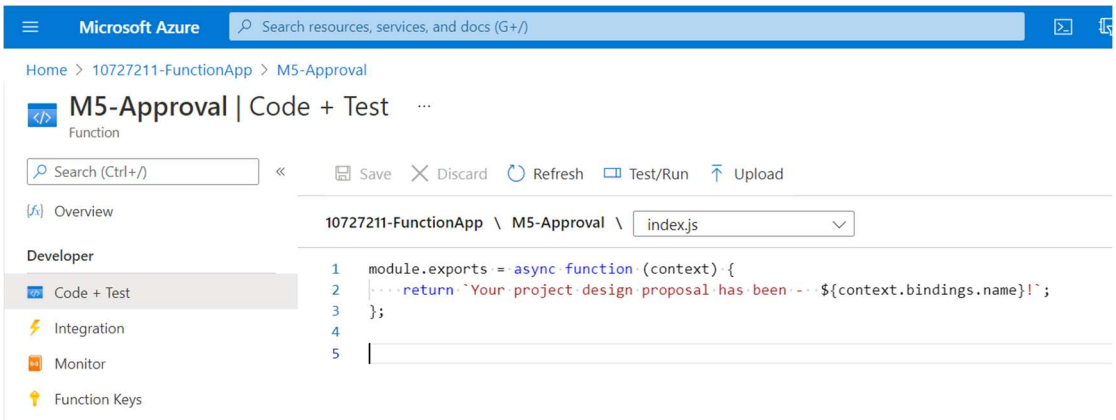
3. Create the orchestrator function



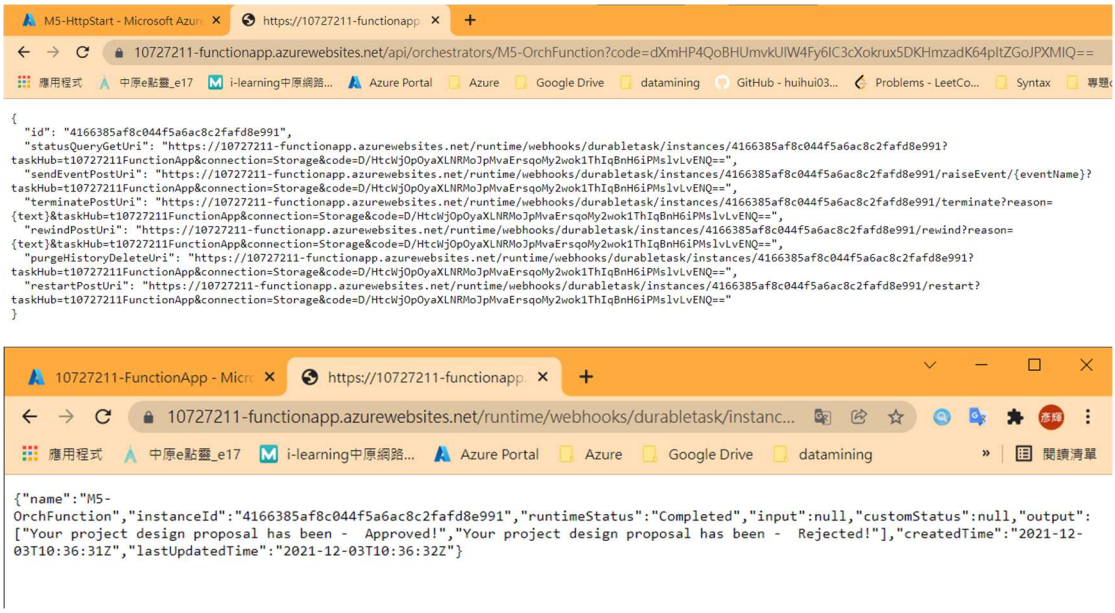


4. Create the activity function

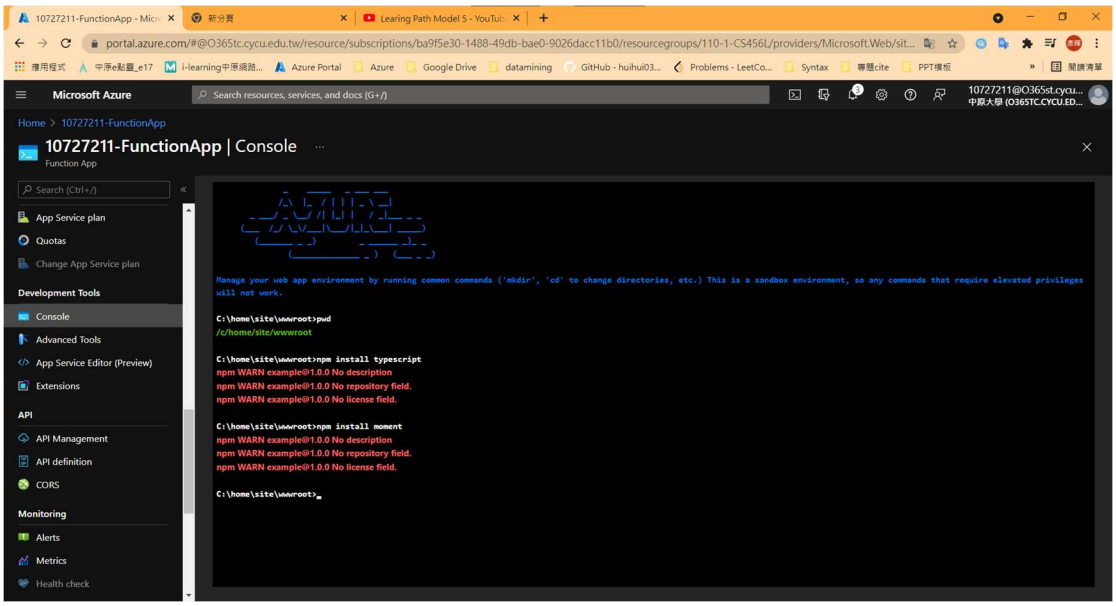




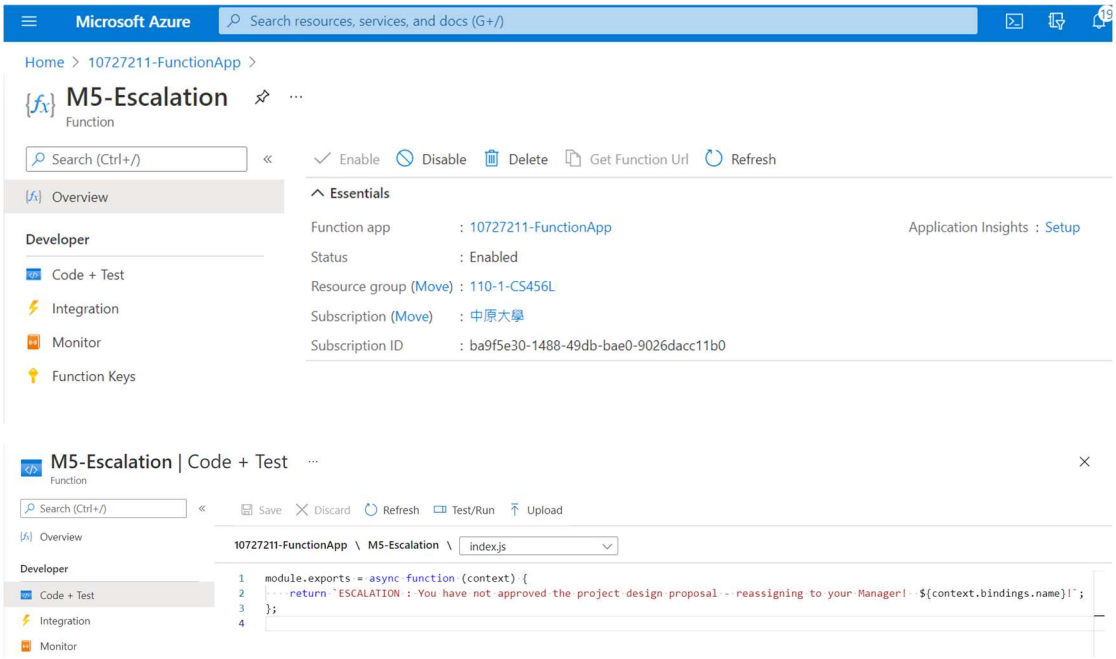
5. Verify that the durable functions workflow starts



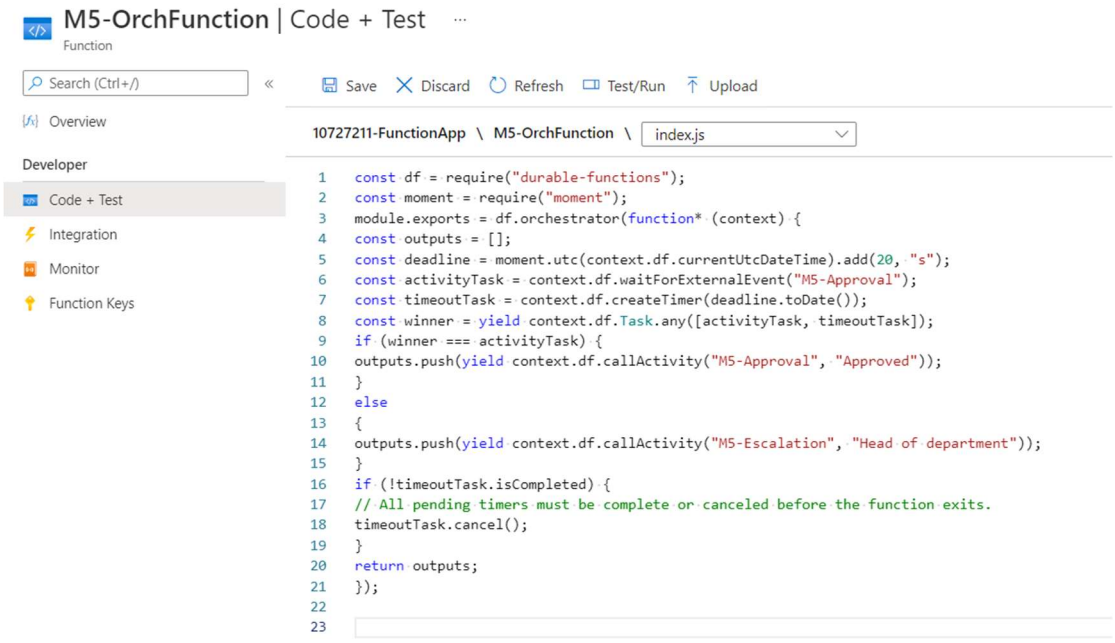
6. Add moment npm package to your function app



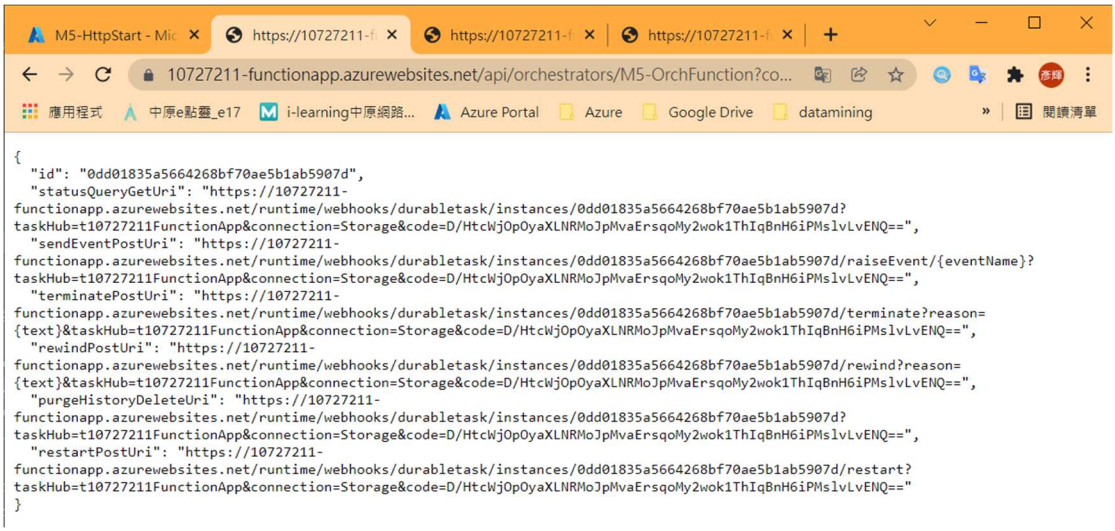
7. Add an escalation activity to your function app

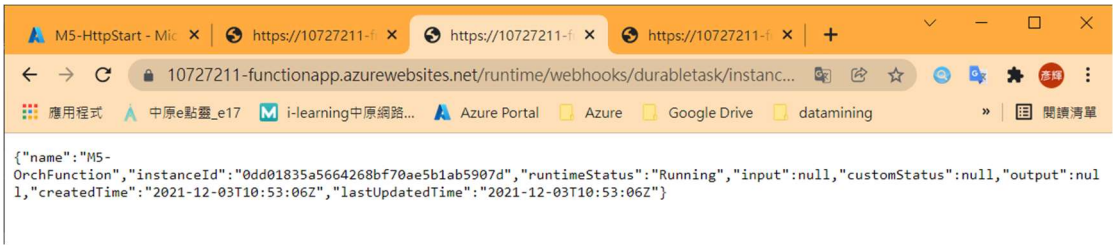


8. Update the orchestration function to use the escalation function

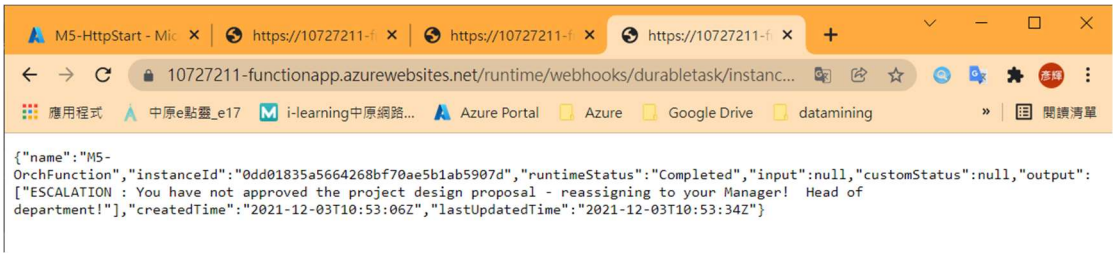


9. Verify that the Durable Functions workflow starts



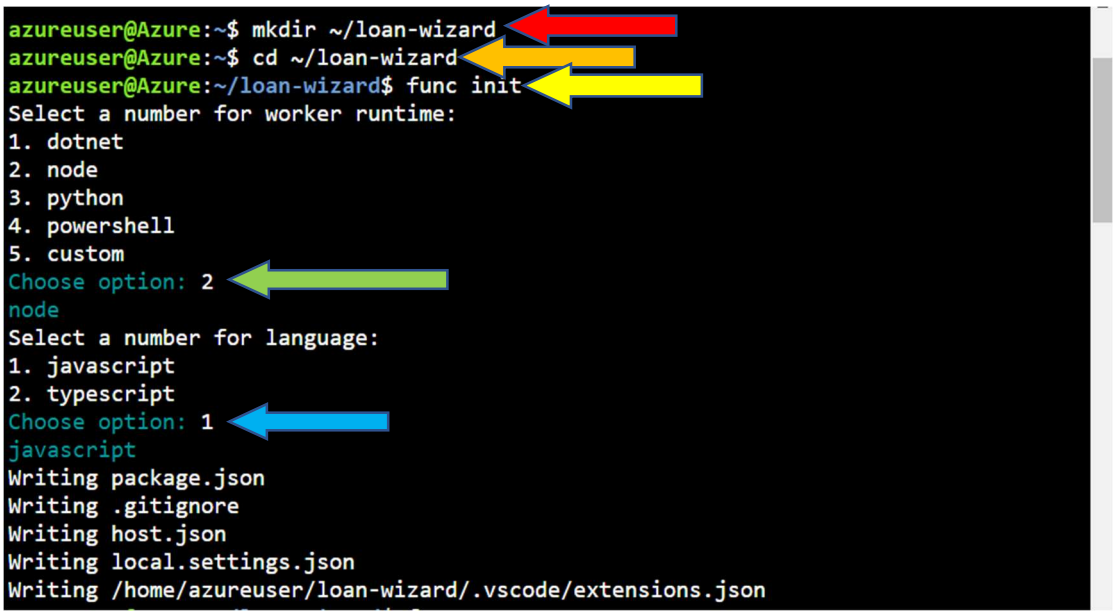


After 20s…

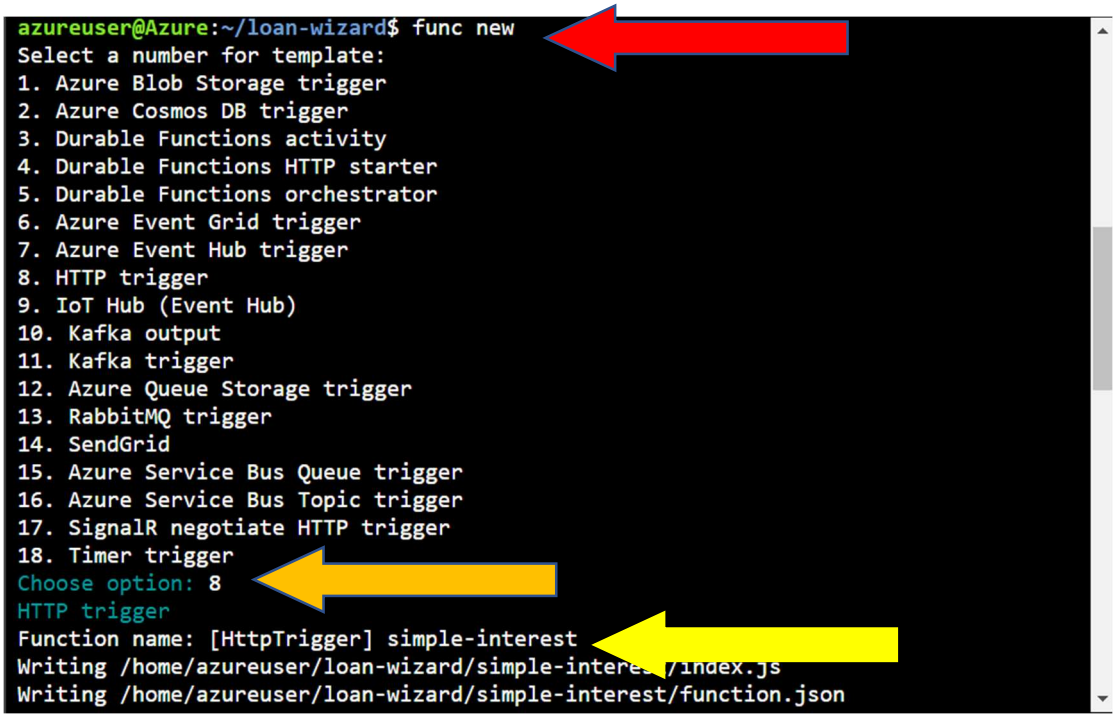


Model 6: Develop, test, and publish Azure Functions by using Azure Functions Core Tools

1. Create a local Azure Functions project

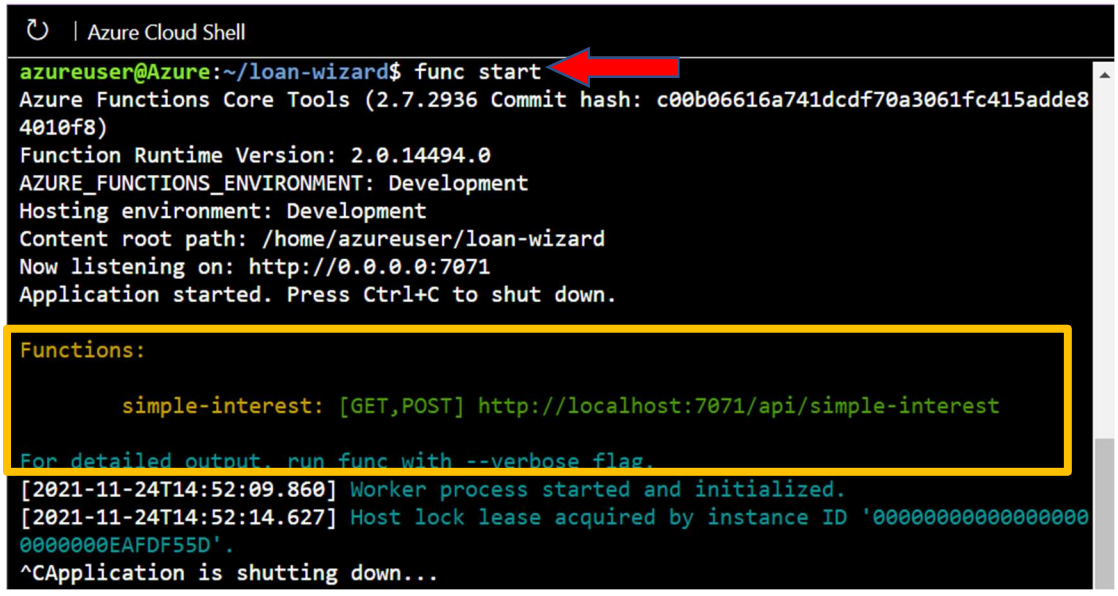


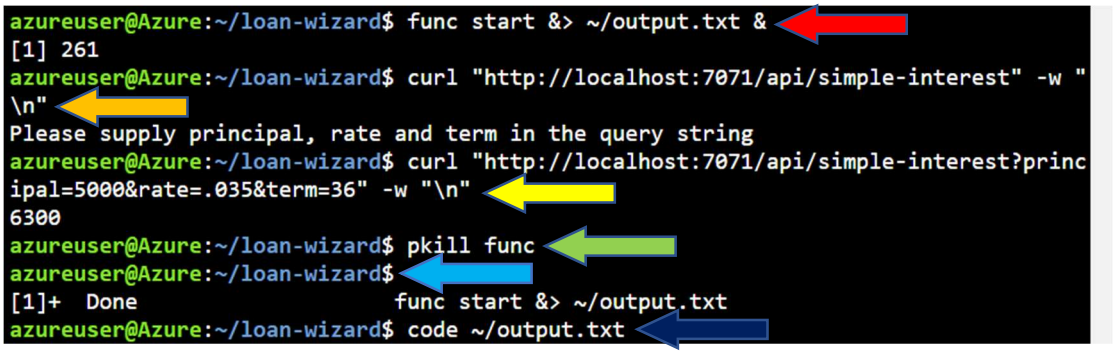
2. Create an HTTP-triggered function



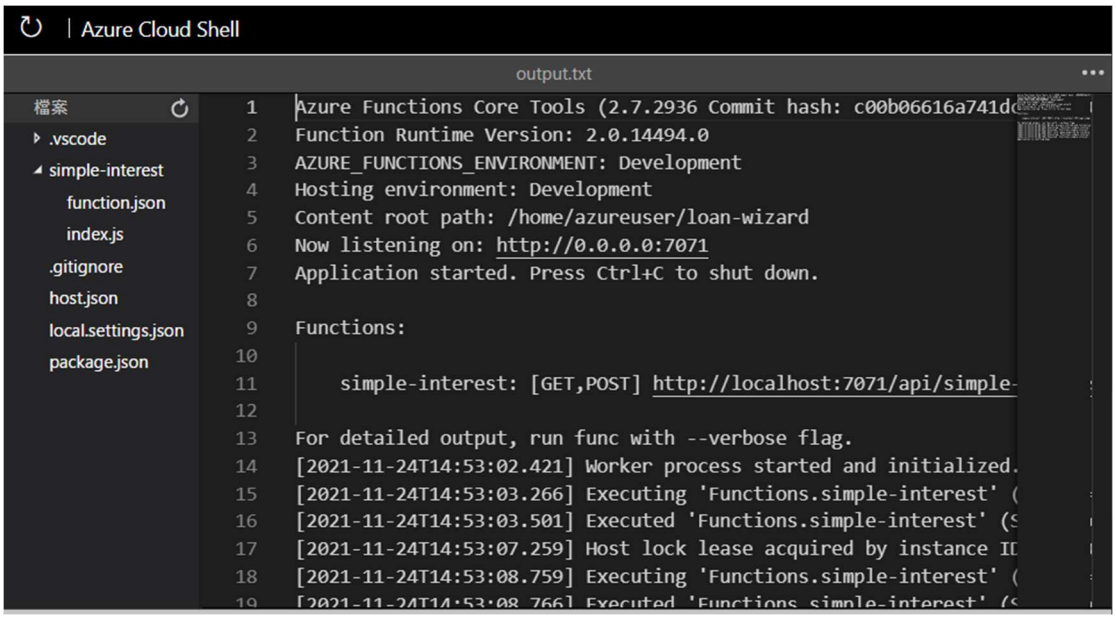
3. Replace index.js

4. Run the function locally





5. See output.txt



6. Create a function app

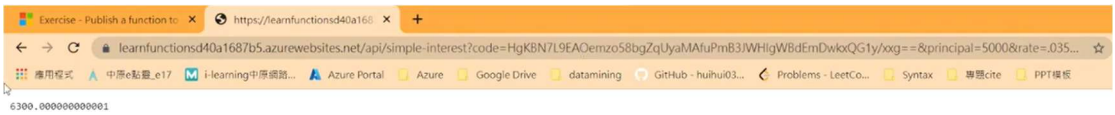
```
azureuser@Azure:~/loan-wizard$ RESOURCEGROUP="learn-0f6b9075-14e1-4f58-9cb9-cd8a0f437
c8a"
azureuser@Azure:~/loan-wizard$ STORAGEACCT=learnstorage$(openssl rand -hex 5)
azureuser@Azure:~/loan-wizard$ FUNCTIONAPP=learnfunctions$(openssl rand -hex 5)
azureuser@Azure:~/loan-wizard$
azureuser@Azure:~/loan-wizard$ az storage account create \
>   --resource-group "$RESOURCEGROUP" \
>   --name "$STORAGEACCT" \
>   --kind StorageV2 \
>   --location centralus
```

```
azureuser@Azure:~/loan-wizard$ az functionapp create \
>   --resource-group "$RESOURCEGROUP" \
>   --name "$FUNCTIONAPP" \
>   --storage-account "$STORAGEACCT" \
>   --runtime node \
>   --consumption-plan-location centralus \
>   --functions-version 3
```

7. Publish to Azure

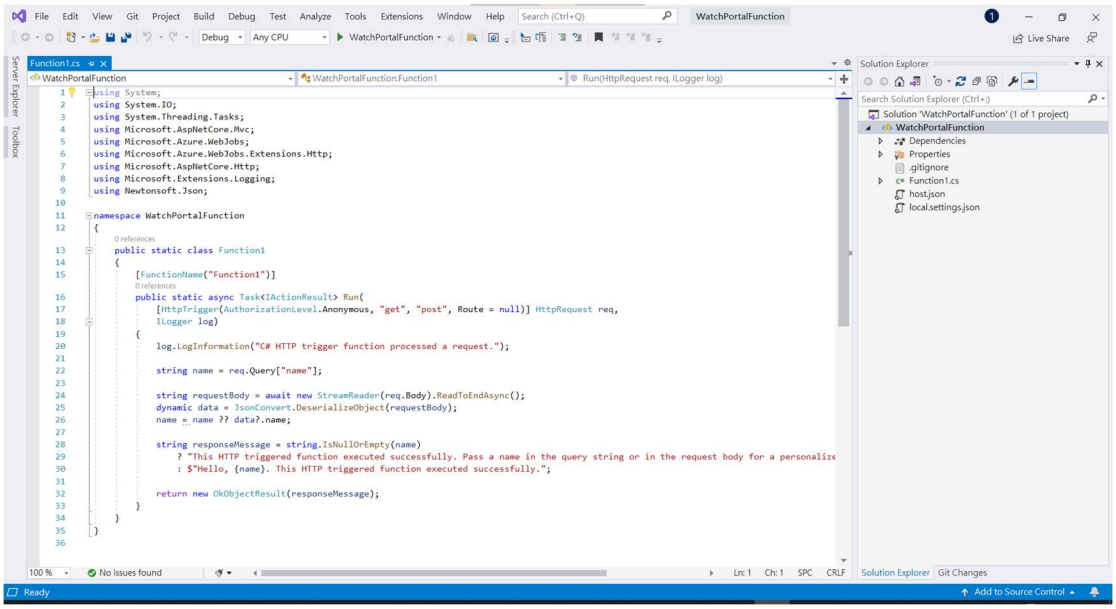
```
azureuser@Azure:~/loan-wizard$ cd ~/loan-wizard
azureuser@Azure:~/loan-wizard$ func azure functionapp publish "$FUNCTIONAPP"
You're trying to publish to a non-v2 function app from v2 tooling.
You can pass --force to force update the app to v2, or switch to v1 or v3 tooling for
 publishing
azureuser@Azure:~/loan-wizard$ func azure functionapp publish "$FUNCTIONAPP" --force
Getting site publishing info...
Creating archive for current directory...
Uploading 1.35 KB [##############################################################]
Upload completed successfully.
Deployment completed successfully.
Syncing triggers...
Functions in learnfunctionsd40a1687b5:
    simple-interest - [httpTrigger]
        Invoke url: https://learnfunctionsd40a1687b5.azurewebsites.net/api/simple-int
erest?code=HgKBN7L9EAOemzo58bgZqUyaMAfuPmB3JWHlgWBdEmDwkxQG1y/xxg==
```

8. Run the function

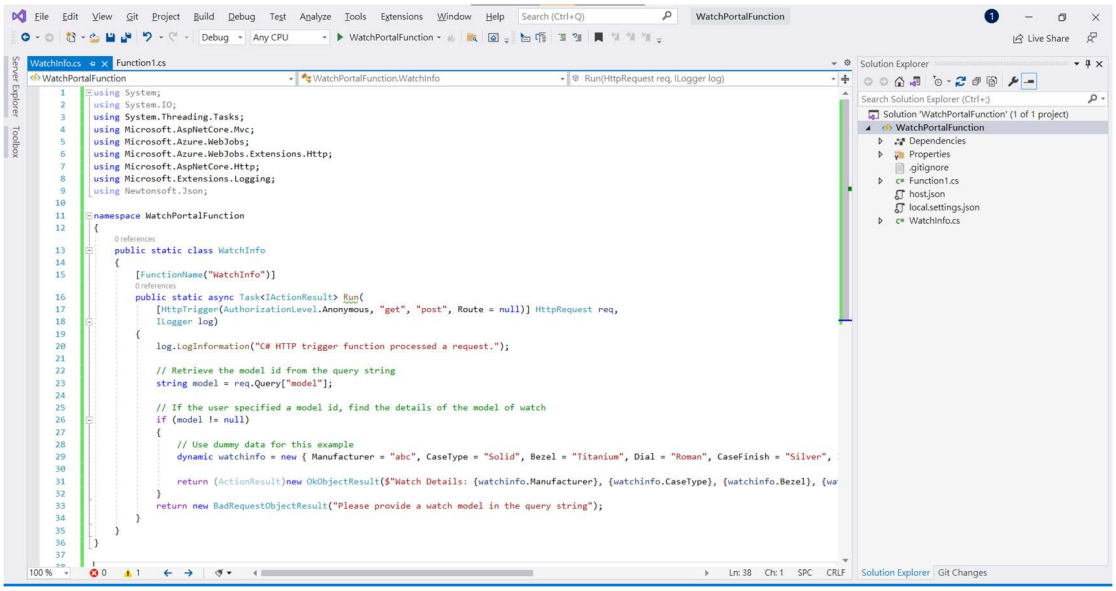


(Model 7: Develop, test, and deploy an Azure Function with Visual Studio

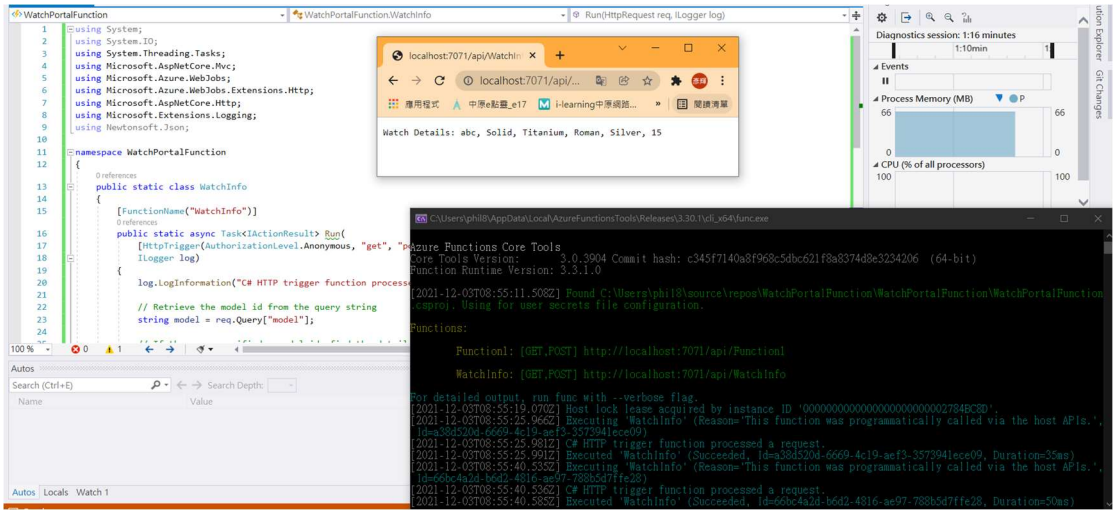
1. Create an Azure Function app



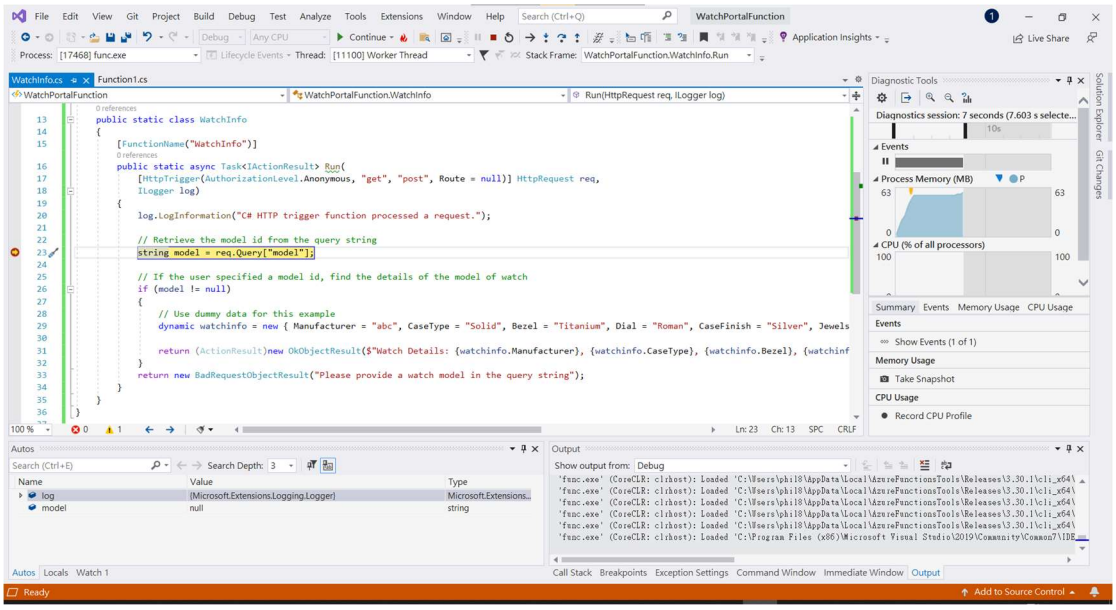
2. Create WatchInfo Function



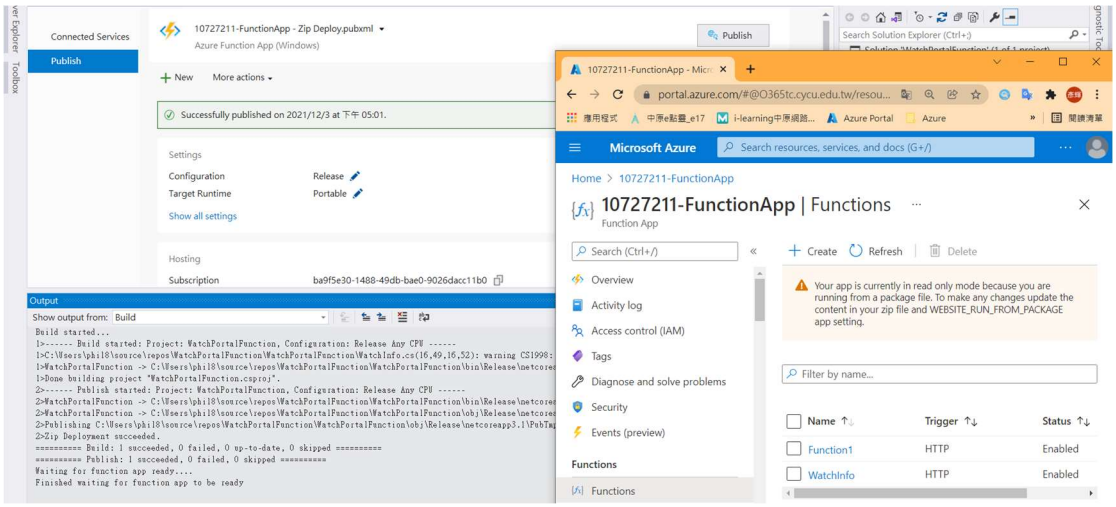
3. Test the Azure Function locally



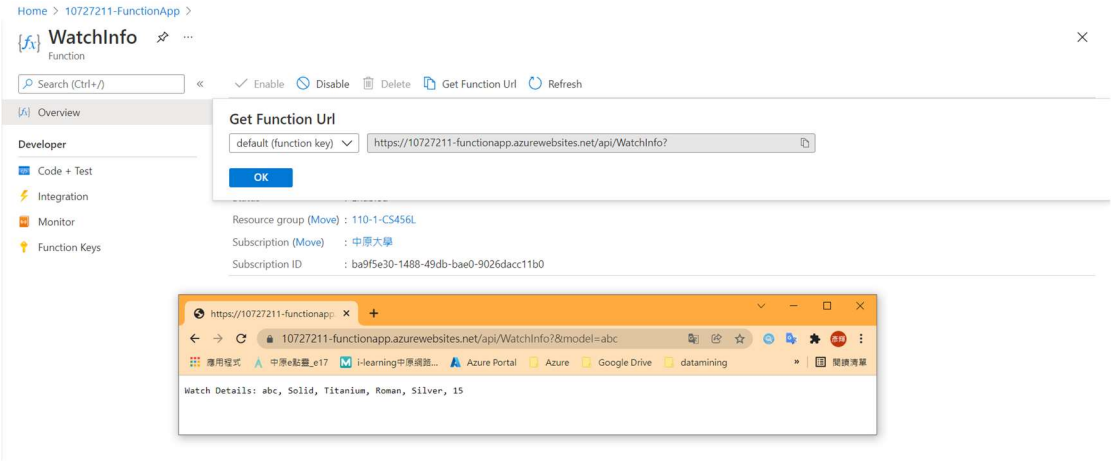
4. Add breakpoint & Test Azure Function locally



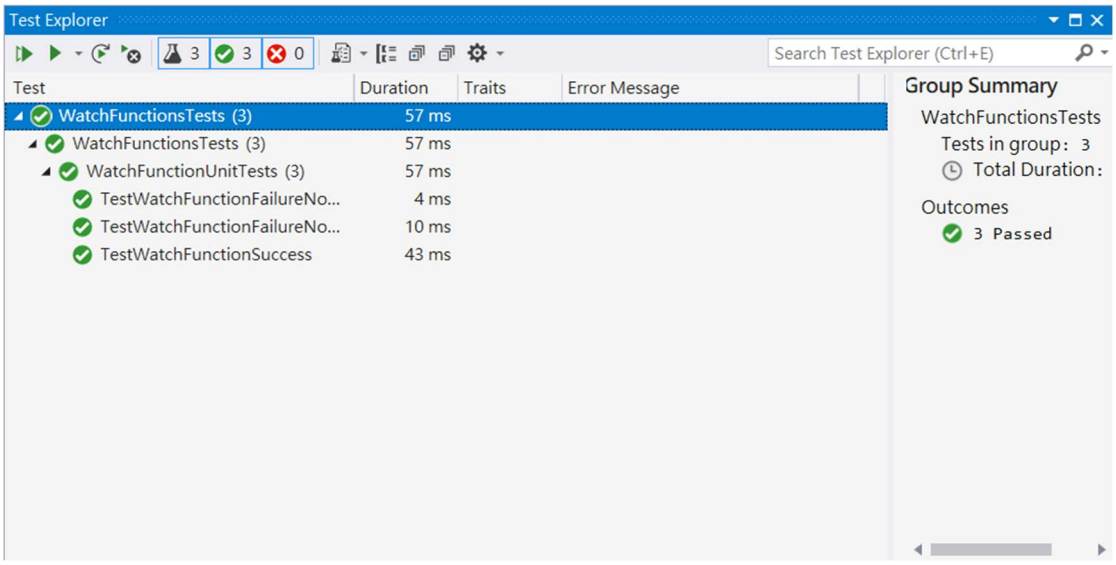
5. Deploy the WatchInfo function to the Azure Function App



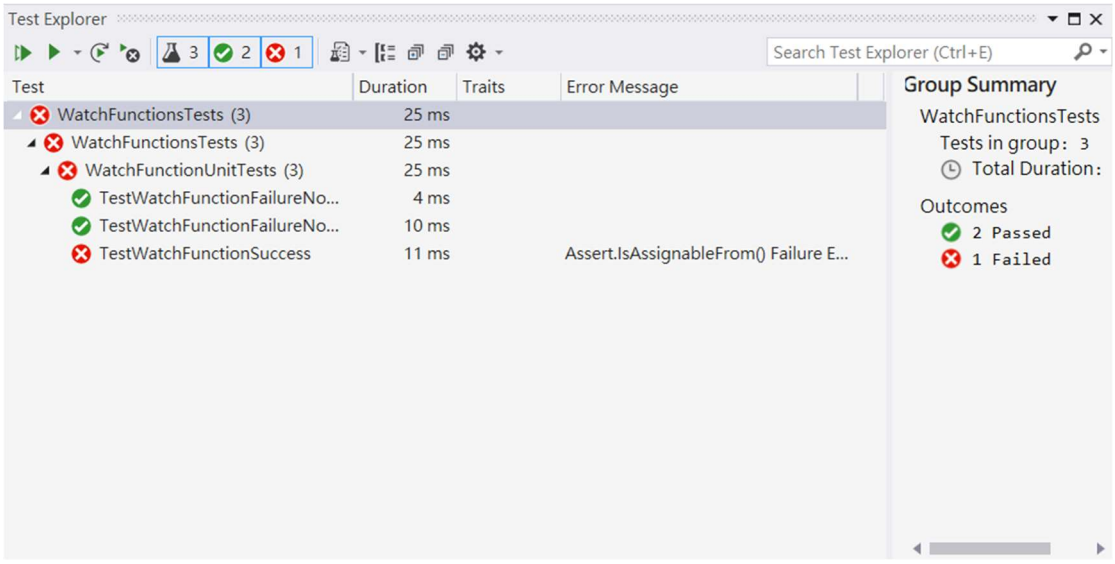
6. Verify the functions have been deployed



7. Unit test an Azure Function (success version)

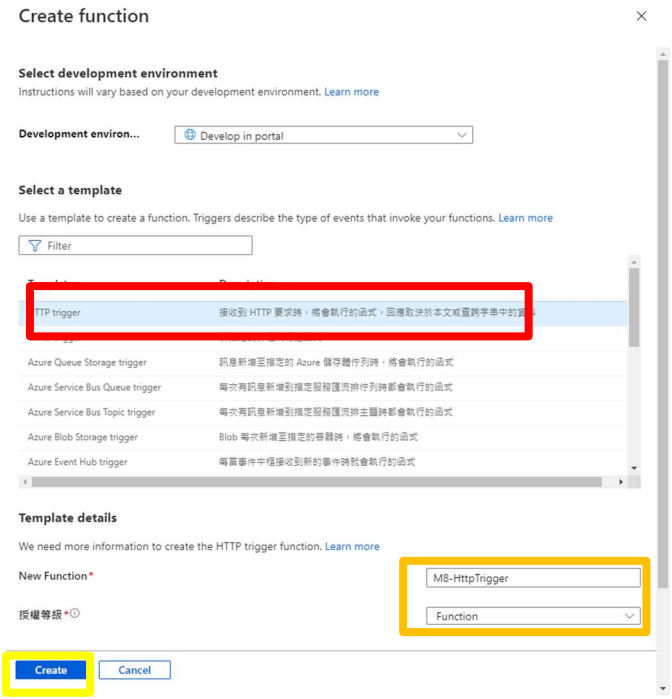


8. Unit test an Azure Function (error version)

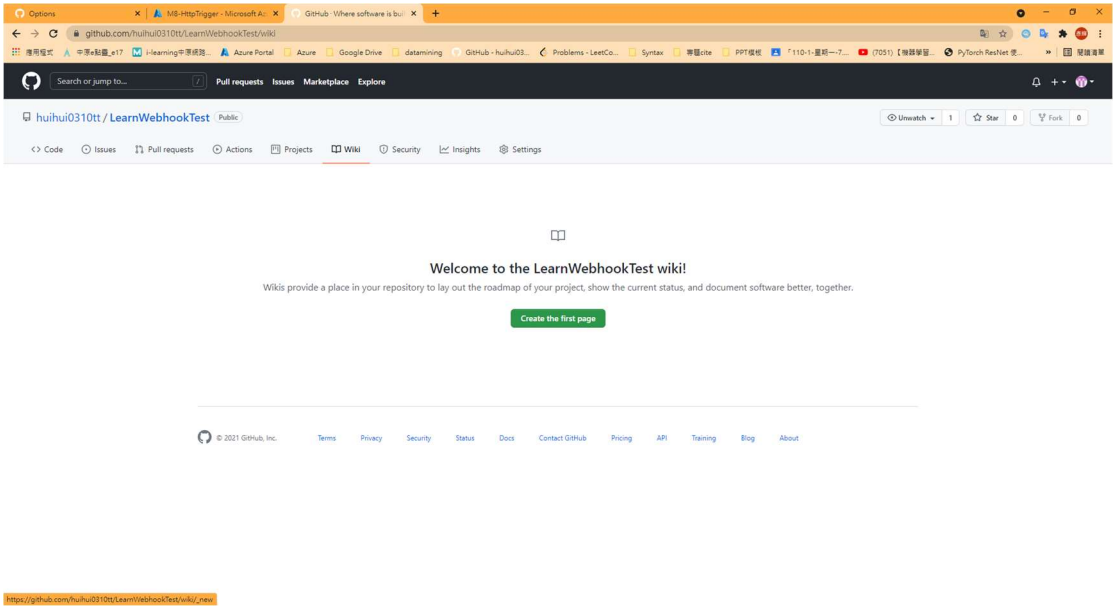


Model 8: Monitor GitHub events by using a webhook with Azure Functions

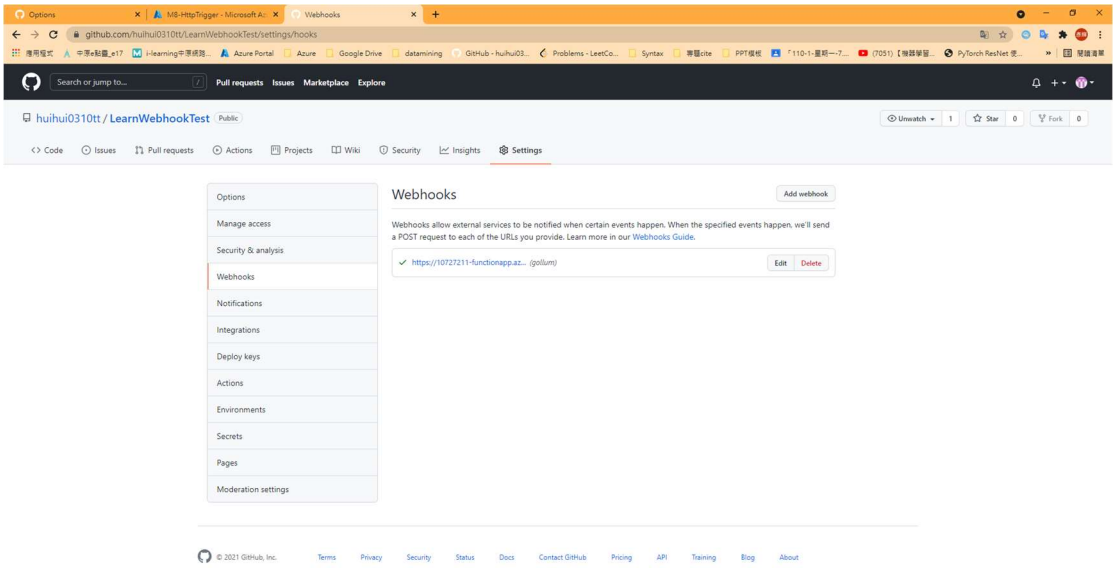
1. Create a Webhook Trigger function



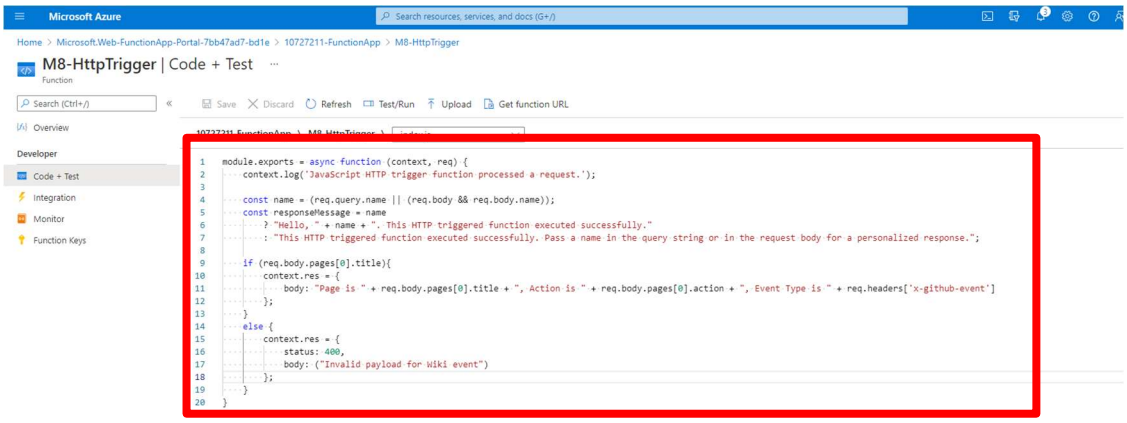
2. Set up a webhook for a GitHub repository



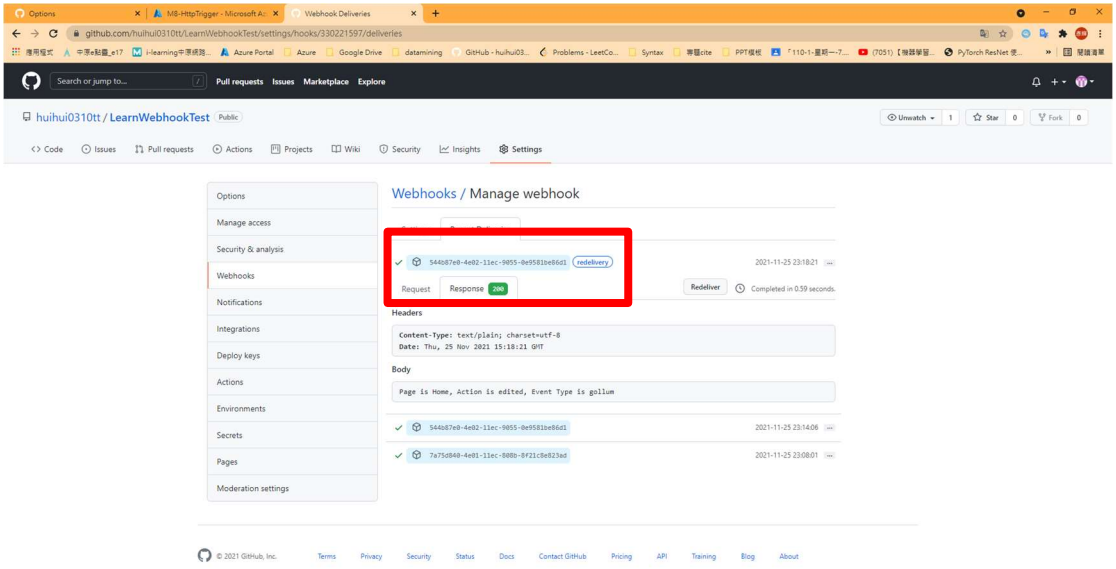
3. Add a webhook and Test



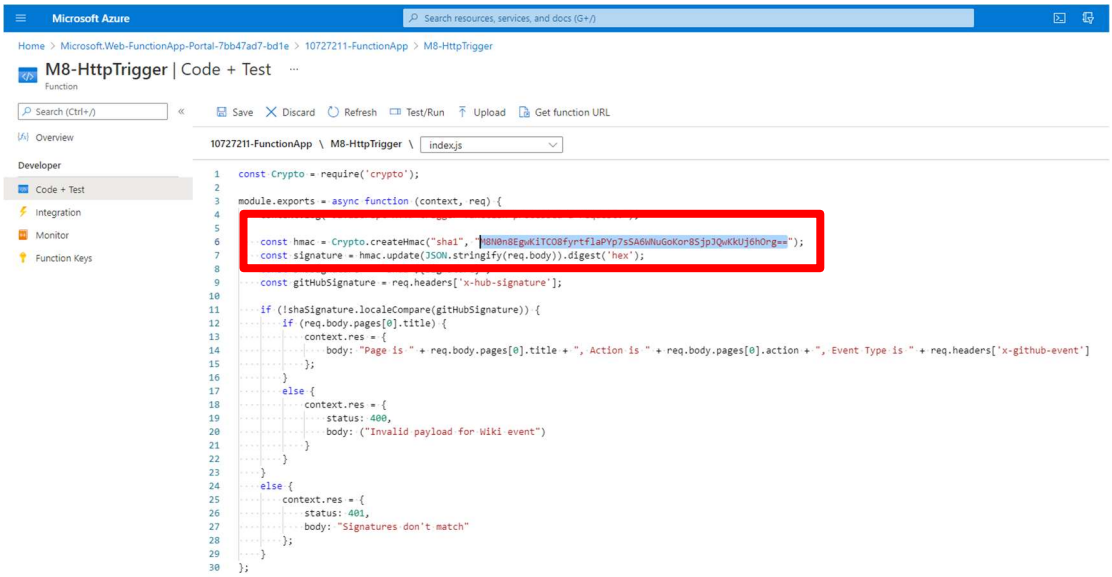
4. Replace index.js in Azure Function

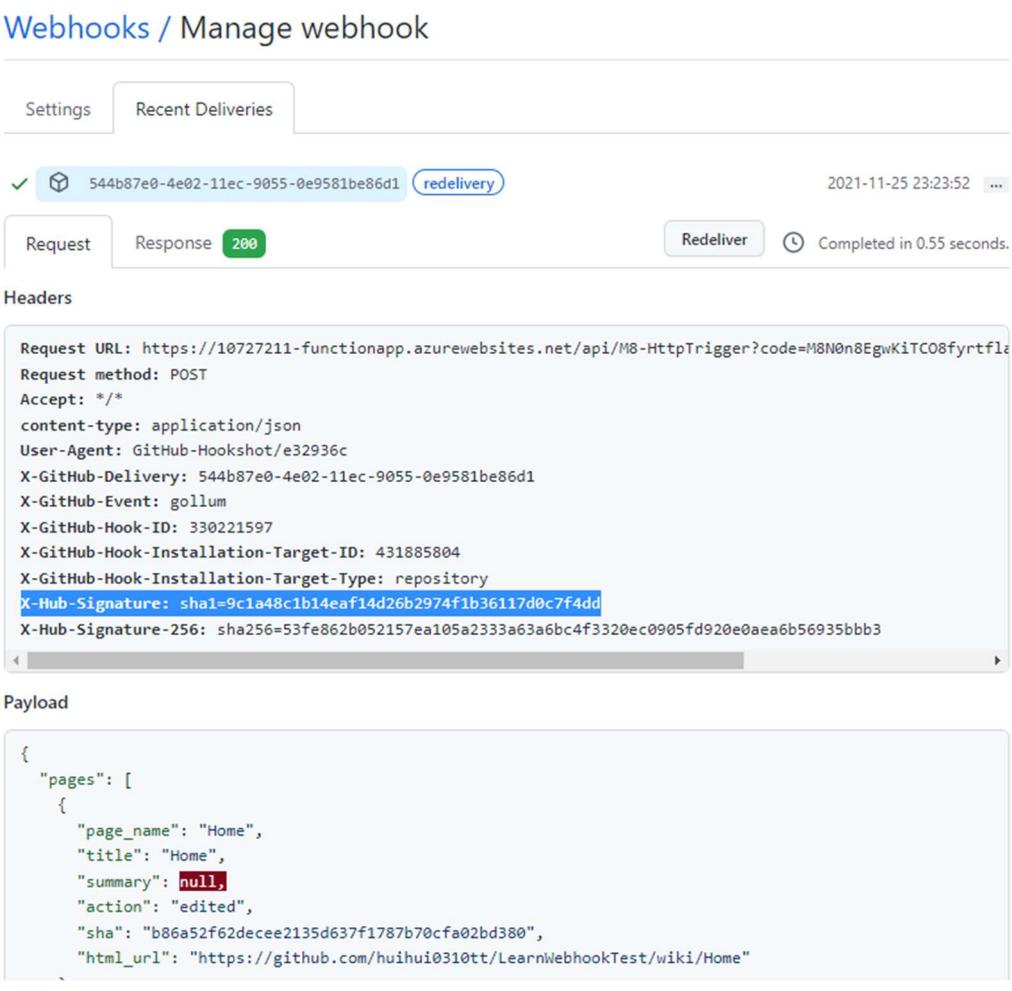


5. 2nd Trigger Webhook



6. Webhook cryption





Model 9: Enable automatic updates in a web application using Azure Functions and SignalR Service

1. Create Resource in Azure

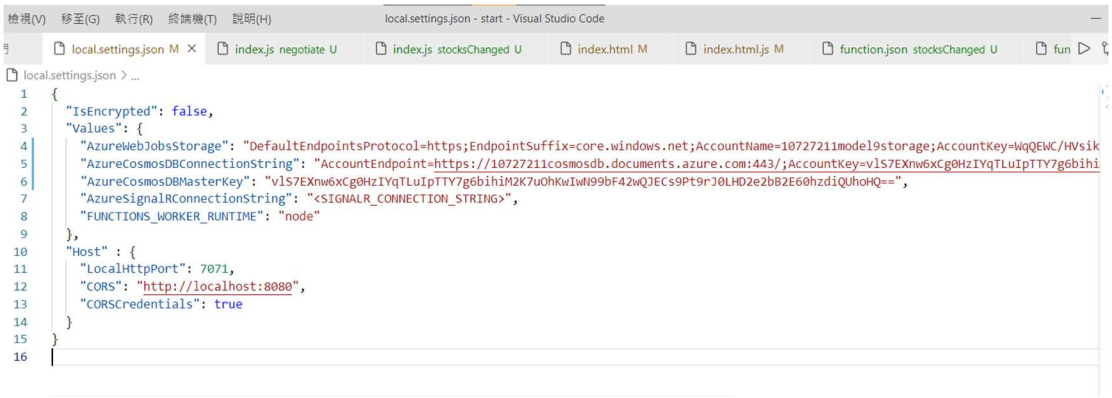
```
export STORAGE_ACCOUNT_NAME=10727211model9storage
echo "Storage Account Name: $STORAGE_ACCOUNT_NAME"

az storage account create \
    --name $STORAGE_ACCOUNT_NAME \
    --resource-group 110-1-CS456L \
    --kind StorageV2 \
    --sku Standard_LRS

az cosmosdb create    \
    --name 10727211cosmosdb \
    --resource-group 110-1-CS456L
```

2. Download sample app code & switch directory

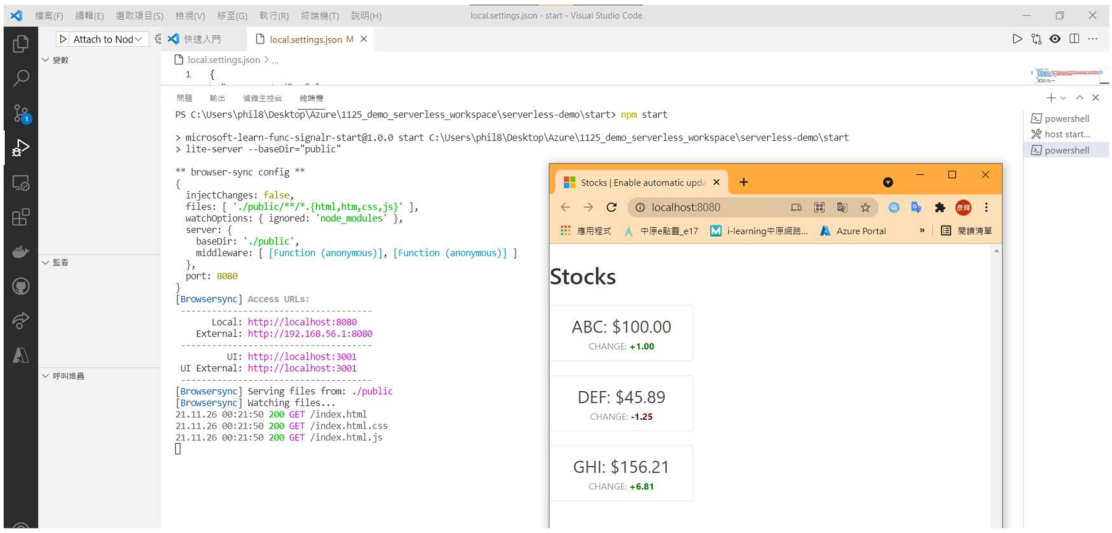
3. Update local settings

```
STORAGE_CONNECTION_STRING=$(az storage account show-connection-string \
--name $(az storage account list \
    --resource-group 110-1-CS456L \
    --query [0].name -o tsv) \
--resource-group 110-1-CS456L \
--query "connectionString" -o tsv)

COSMOSDB_ACCOUNT_NAME=$(az cosmosdb list \
        --resource-group 110-1-CS456L \
        --query [0].name -o tsv)
```

```
COSMOSDB_CONNECTION_STRING=$(az cosmosdb list-connection-strings    \
    --name $COSMOSDB_ACCOUNT_NAME \
    --resource-group 110-1-CS456L \
    --query "connectionStrings[?description=='Primary SQL Connection
String'].connectionString" -o tsv)

COSMOSDB_MASTER_KEY=$(az cosmosdb list-keys \
--name $COSMOSDB_ACCOUNT_NAME \
--resource-group 110-1-CS456L \
--query primaryMasterKey -o tsv)

printf "\n\nReplace <STORAGE_CONNECTION_STRING>
with:\n$STORAGE_CONNECTION_STRING\n\nReplace
<COSMOSDB_CONNECTION_STRING>
with:\n$COSMOSDB_CONNECTION_STRING\n\nReplace
<COSMOSDB_MASTER_KEY> with:\n$COSMOSDB_MASTER_KEY\n\n"
```



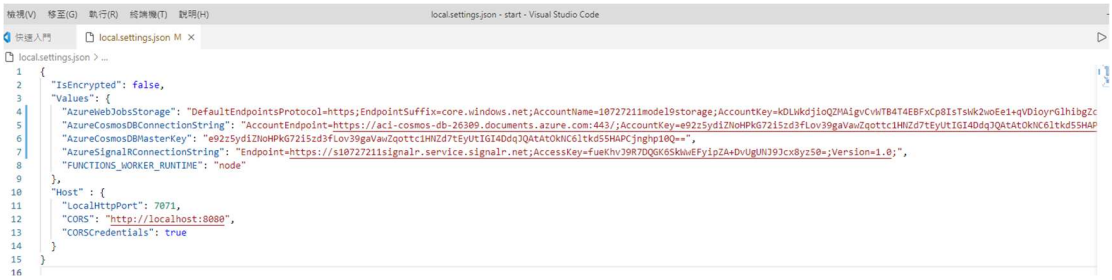
4. Run the application



1. Create a SignalR account

```
SIGNALR_SERVICE_NAME=s10727211signalr
az signalr create \
    --name $SIGNALR_SERVICE_NAME \
    --resource-group 110-1-CS456L \
    --sku Free_DS2 \
    --unit-count 1
az resource update \
    --resource-type Microsoft.SignalRService/SignalR \
    --name $SIGNALR_SERVICE_NAME \
    --resource-group 110-1-CS456L \
    --set properties.features[flag=ServiceMode].value=Serverless
SIGNALR_CONNECTION_STRING=$(az signalr key list \
    --name $(az signalr list \
        --resource-group 110-1-CS456L \
        --query [0].name -o tsv) \
    --resource-group 110-1-CS456L \
    --query primaryConnectionString -o tsv)
printf "\n\nReplace <SIGNALR_CONNECTION_STRING>
with:\n$SIGNALR_CONNECTION_STRING\n\n"
```

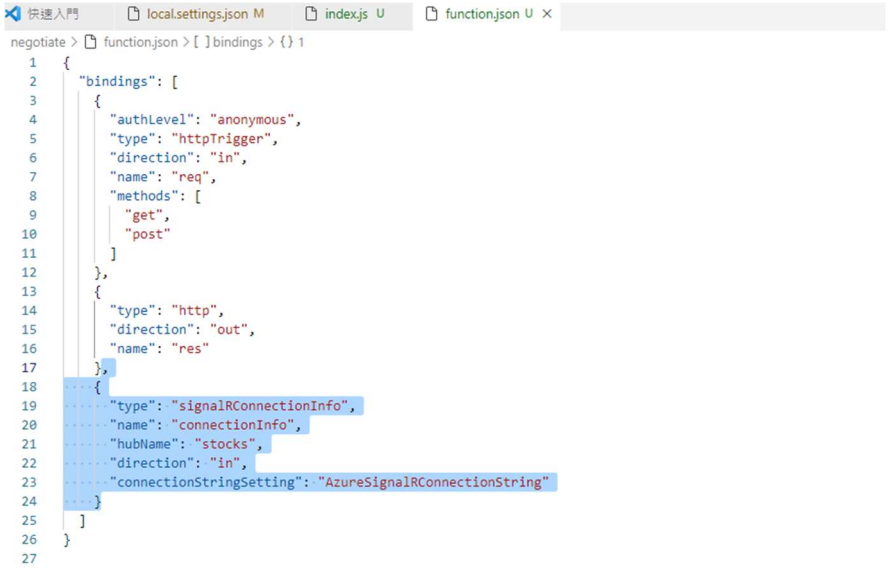
2. Update local settings



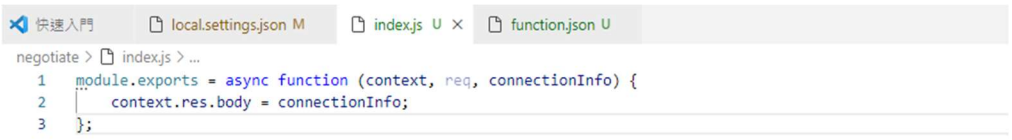
3. Create Azure Functions

	Azure Functions: Create Function
Template	HTTP Trigger
Name	negotiate
Authorization level	Anonymous

4. Update client connections : negotiate > function.json



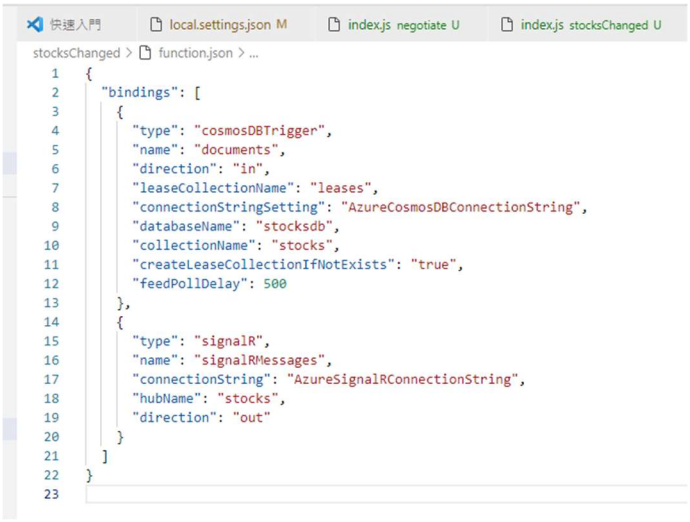
5. Update client connections : negotiate > index.js



6. Create Azure Functions

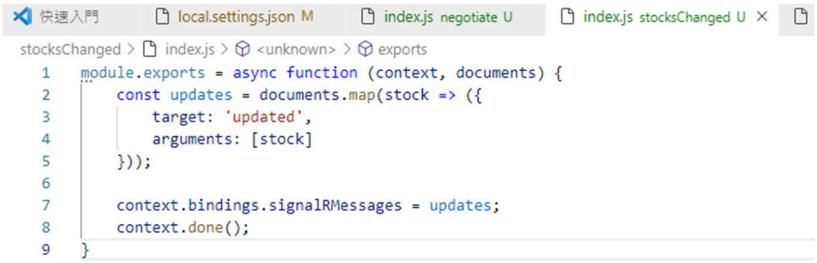
	Azure Functions: Create Function
Template	Azure Cosmos DB Trigger
Name	stocksChanged
App setting for your Azure Cosmos DB account	AzureCosmosDBConnectionString
Database name	stocksdb
Collection name	stocks
Collection name for leases	leases
Create lease collection if not exists	true

7. Update Detect and broadcast database changes : stocksChanged > function.json



```
{
  "bindings": [
    {
      "type": "cosmosDBTrigger",
      "name": "documents",
      "direction": "in",
      "leaseCollectionName": "leases",
      "connectionStringSetting": "AzureCosmosDBConnectionString",
      "databaseName": "stocksdb",
      "collectionName": "stocks",
      "createLeaseCollectionIfNotExists": "true",
      "feedPollDelay": 500
    },
    {
      "type": "signalR",
      "name": "signalRMessages",
      "connectionString": "AzureSignalRConnectionString",
      "hubName": "stocks",
      "direction": "out"
    }
  ]
}
```

8. Update Detect and broadcast database changes : stocksChanged > index.js

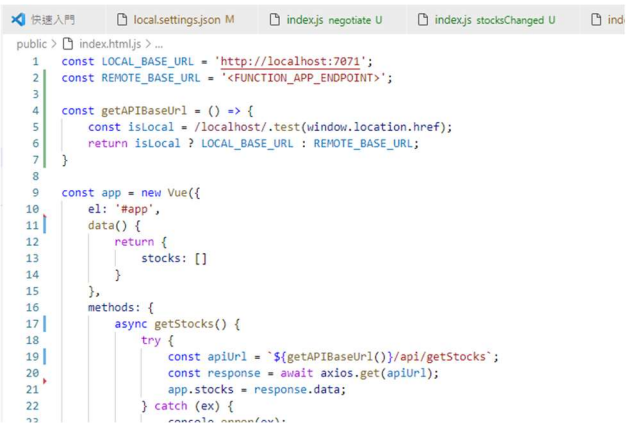


```
module.exports = async function (context, documents) {
    const updates = documents.map(stock => ({
        target: 'updated',
        arguments: [stock]
    }));

    context.bindings.signalRMessages = updates;
    context.done();
}
```

9. Update the web application : public > index.html

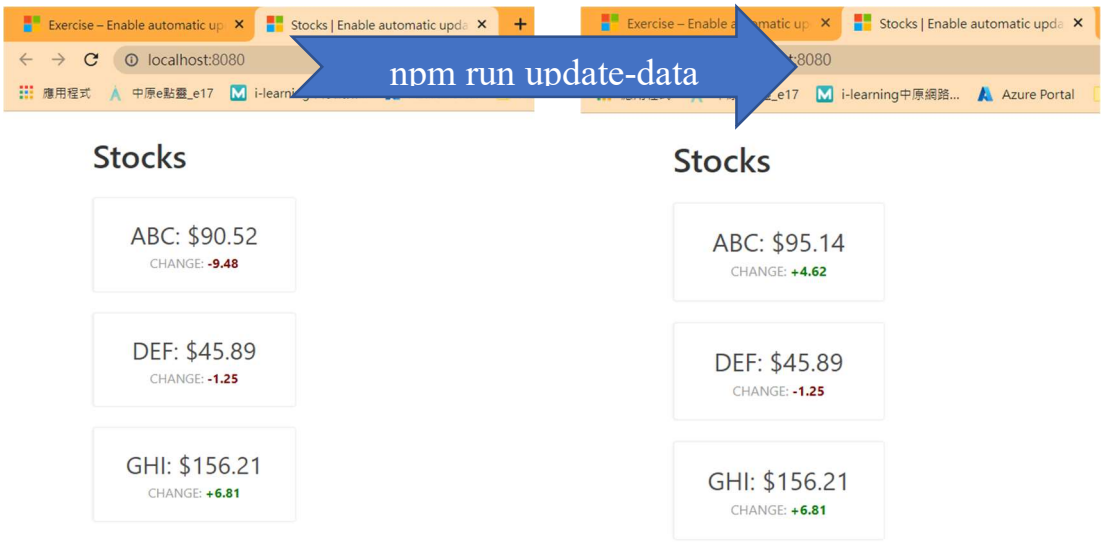
```
<!-- BEGIN: Replace markup in this section -->
<div id="app" class="container">
    <h1 class="title">Stocks</h1>
    <div id="stocks">
        <div v-for="stock in stocks" class="stock">
            <transition name="fade" mode="out-in">
                <div class="list-item" :key="stock.price">
                    <div class="lead">{{ stock.symbol }}: ${{ stock.price }}</div>
                    <div class="change">Change:
                        <span
                            :class="{ 'is-up': stock.changeDirection === '+', 'is-down': stock.changeDirection === '-' }">
                            {{ stock.changeDirection }}{{ stock.change }}
                        </span>
                    </div>
                </div>
            </transition>
        </div>
    </div>
</div>
<!-- END -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.10/vue.min.js" integrity="sha256-chlNFSVx3TdcQ2Xlw7SvnbLAavAQLO0Y/LBiWX04viY=" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.19.0/axios.min.js" integrity="sha256-S1J4GVHHDMiirir9qsXWc8ZWw74PHHafpsHp5PXtjTs=" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@aspnet/signalr@1.1.0/dist/browser/signalr.js"></script>
<script src="index.html.js" type="text/javascript"></script>
</body>
```

10. Update the web application : public > index.html.js



```
const LOCAL_BASE_URL = 'http://localhost:7071';
const REMOTE_BASE_URL = '<FUNCTION_APP_ENDPOINT>';

const getAPIBaseUrl = () => {
    const isLocal = /localhost/.test(window.location.href);
    return isLocal ? LOCAL_BASE_URL : REMOTE_BASE_URL;
}

const app = new Vue({
    el: '#app',
    data() {
        return {
            stocks: []
        }
    },
    methods: {
        async getStocks() {
            try {
                const apiUrl = `${getAPIBaseUrl()}/api/getStocks`;
                const response = await axios.get(apiUrl);
                app.stocks = response.data;
            } catch (ex) {
```

11. Run the application



1. Create Azure Functions

	Azure Functions: Deploy to Function App
Template	Create new Function App in Azure... Advanced
Name	10727211-deployfunctionapp
Running Stack	Node.js 8 LTS
Resource Group	110-1-CS456L
Location for new resource	東南亞 (for CS456L)
Hosting Plan	Consumption
storage	10727211model9storage
Application Insights resource	Skip for now

2. Update the function URL

3. Upload Function Settings

	Azure Functions: Upload local settings
Function App in Azure	10727211-deployfunctionapp

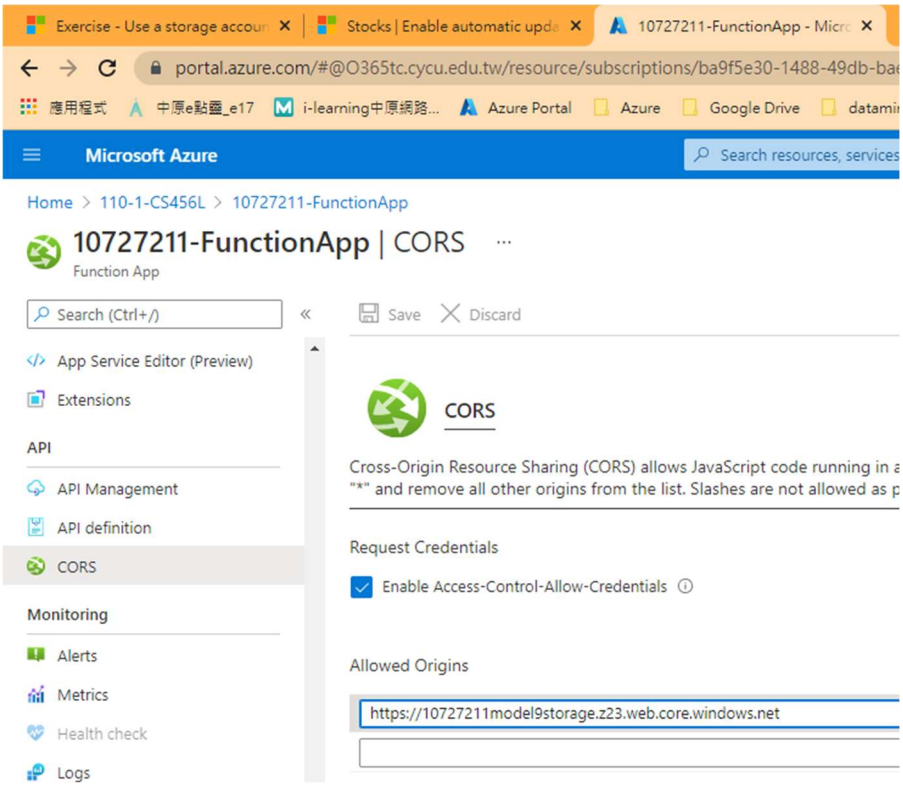
4. Configure static websites in Azure Storage

	Azure Storage: Configure Static Website
Storage account	10727211model9storage
Default file	Index.html
Error document	Index.html

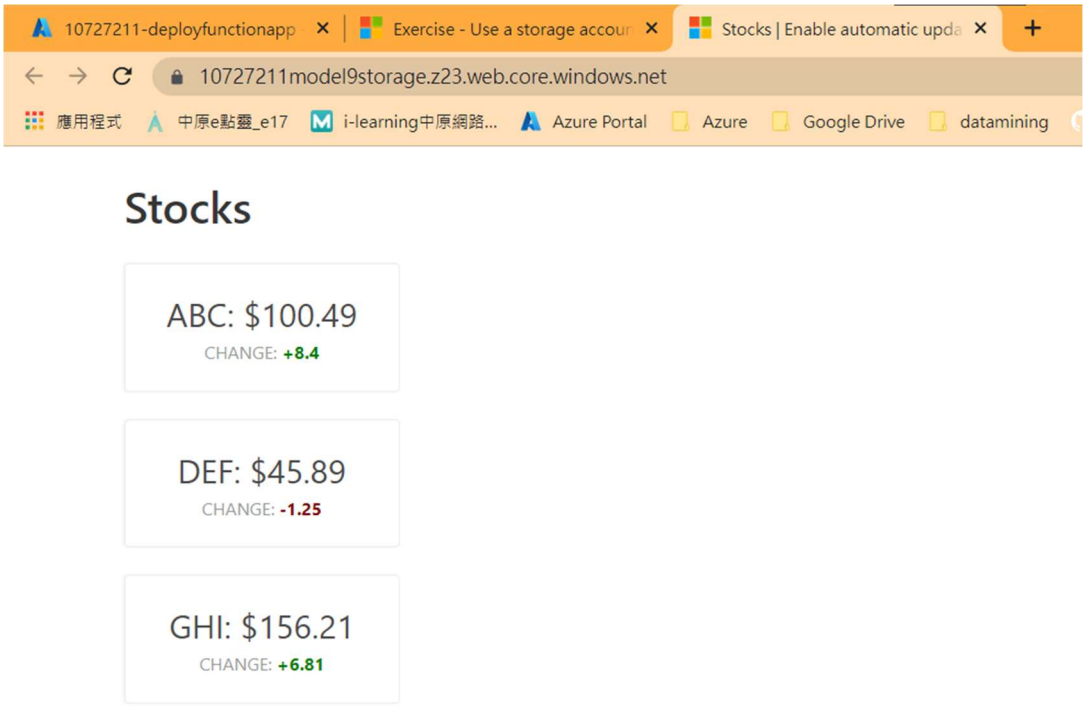
5. Deploy the web application to Azure Storage

	Azure Storage: Deploy to static website via Azure Storage
Storage account	10727211model9storage
Select the folder to deploy	Select **browse** and choose the *public* subfolder containing the web app.

6. Set up CORS in the function app

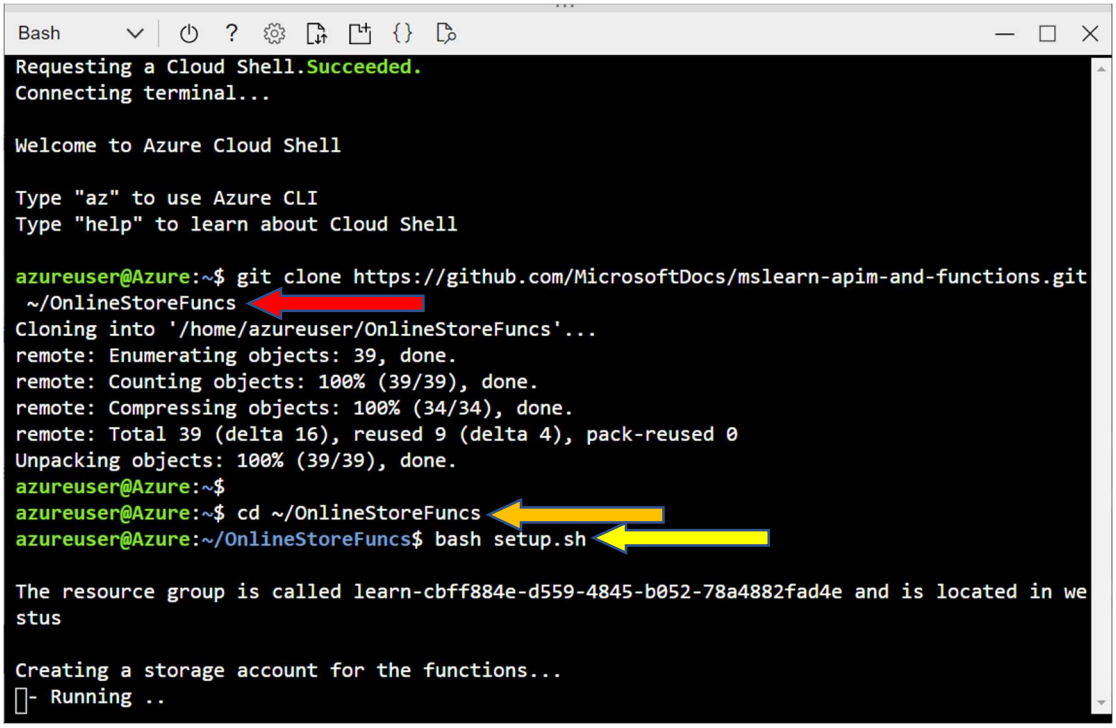


7. Run the deployed application

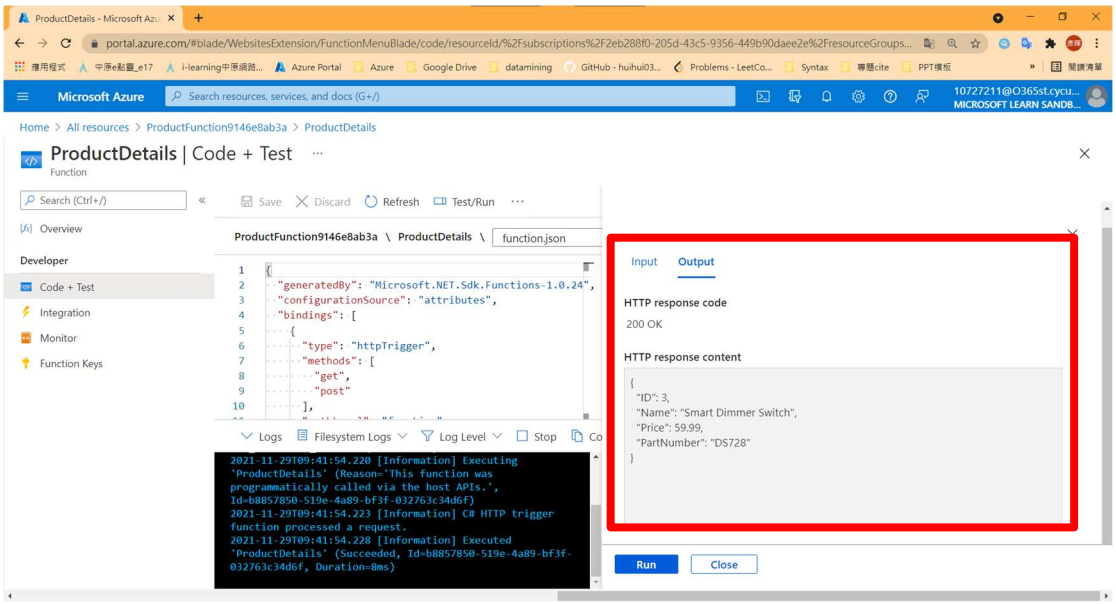


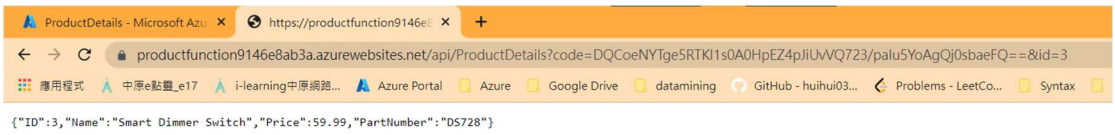
Model 10: Expose multiple Azure Function apps as a consistent API by using Azure API Management

1. Create functions



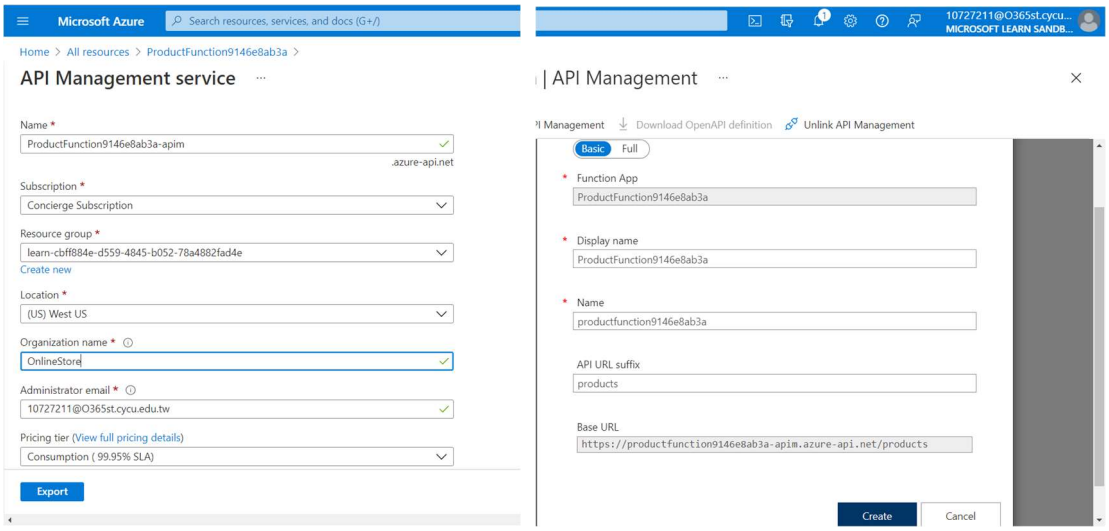
2. Test ProductDetails function



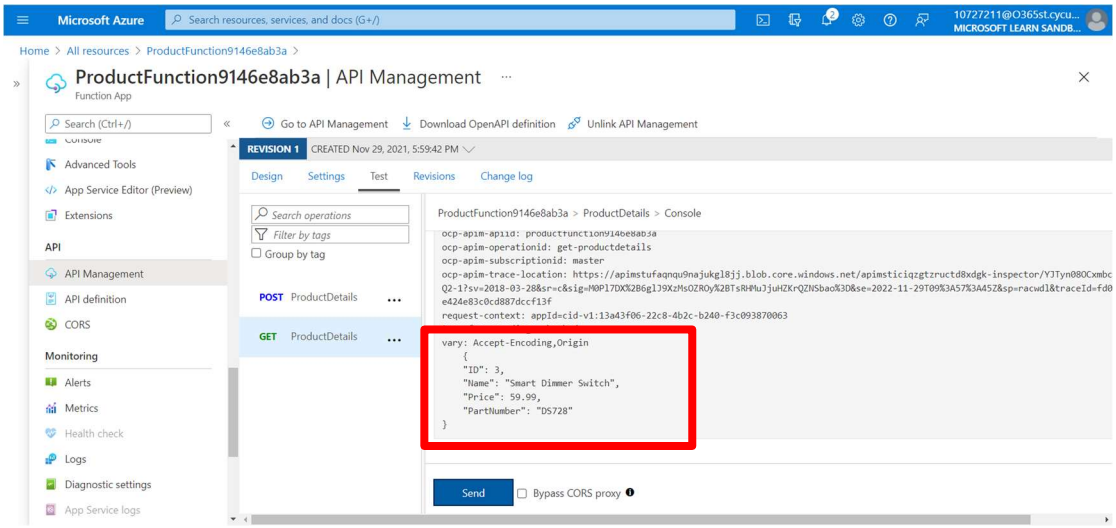


{"ID":3,"Name":"Smart Dimmer Switch","Price":59.99,"PartNumber":"DS728"}

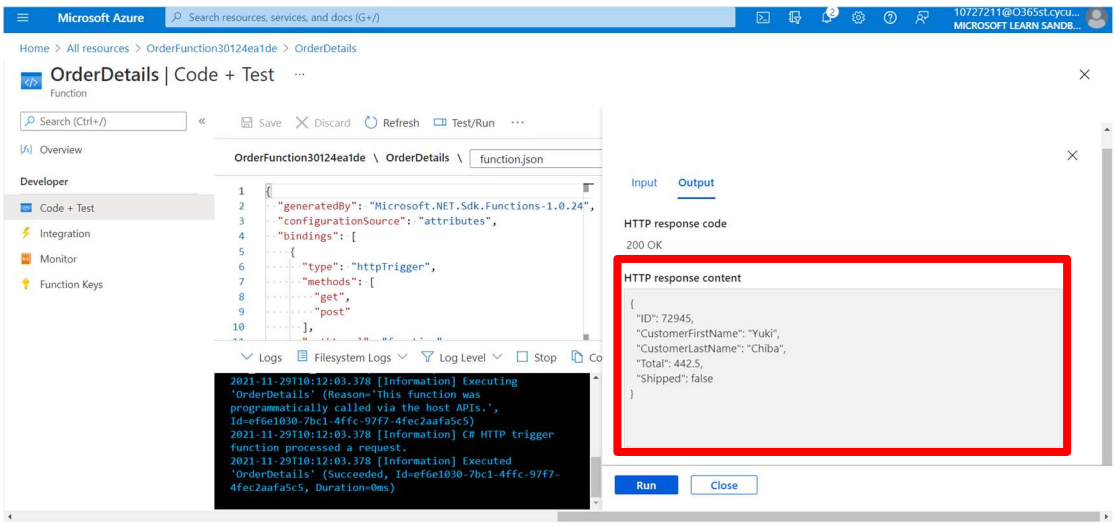
3. Expose function app as an API using Azure API Management



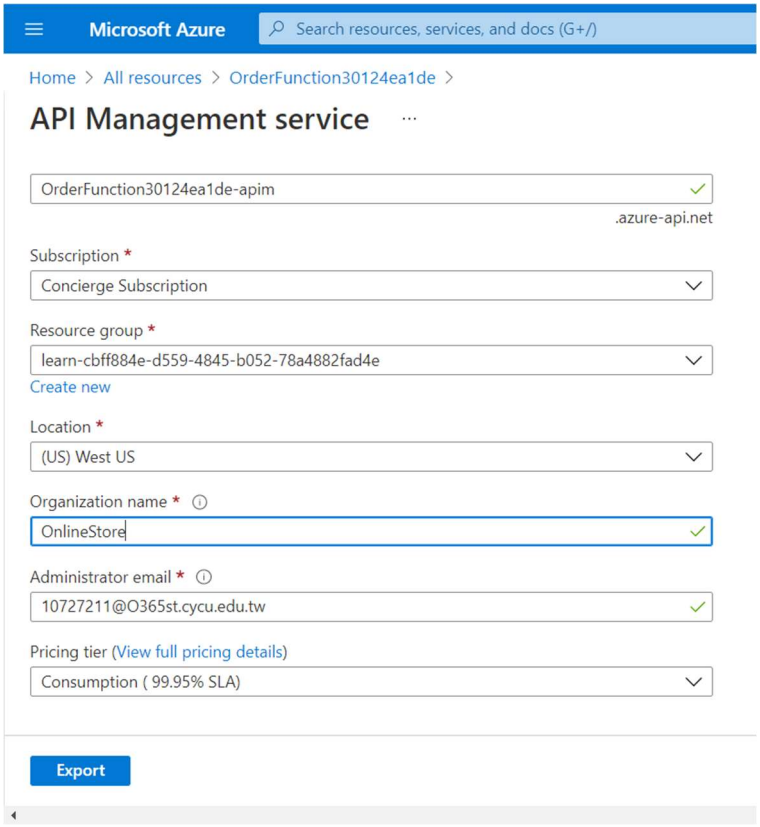
4. Test the OnlineStore products endpoint

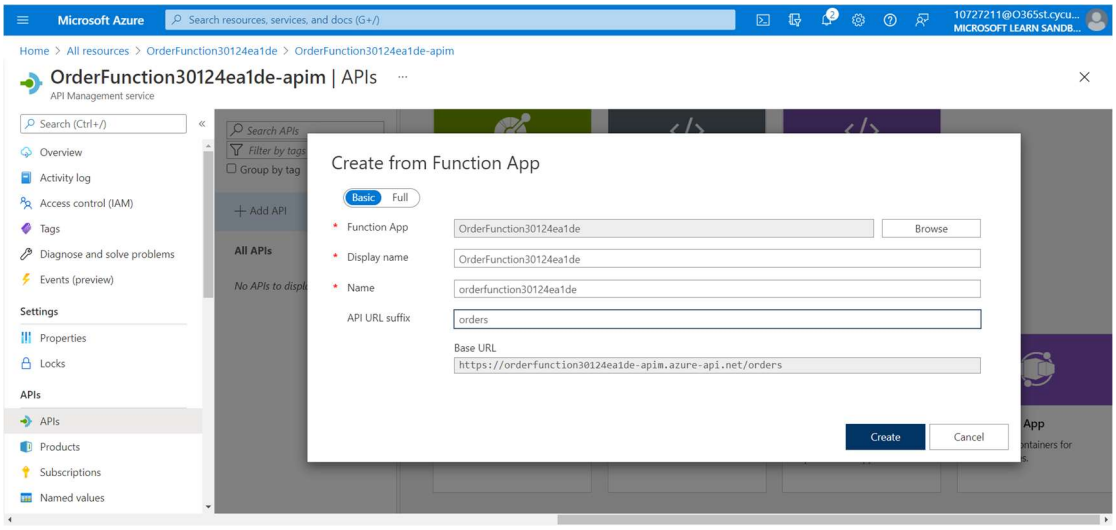


5. Test the OrderDetails function

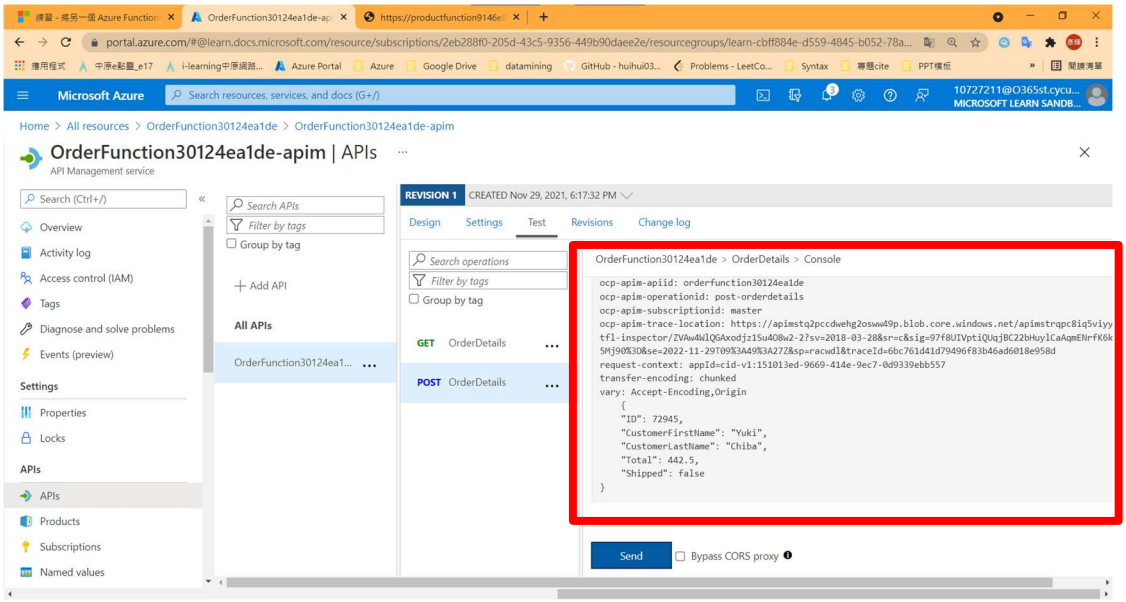


6. Add a function to an existing API

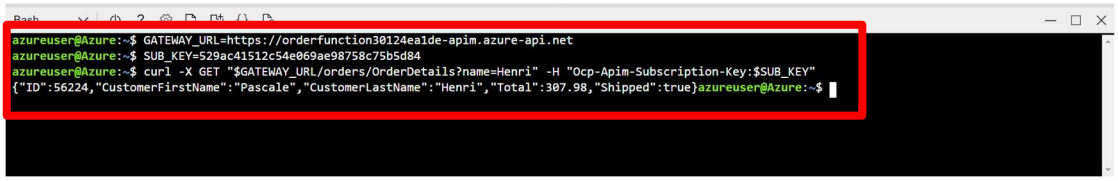


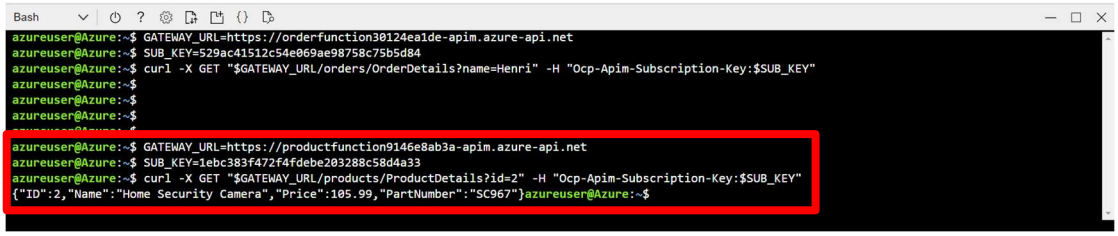


7. Test the OnlineStore orders endpoint in the portal



8. Test the combined API

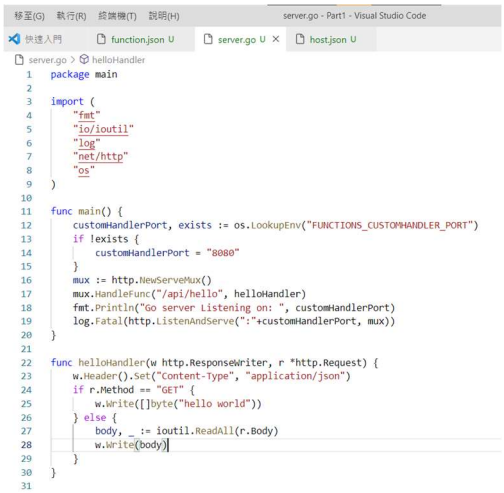




Model 11: Build serverless apps with Go

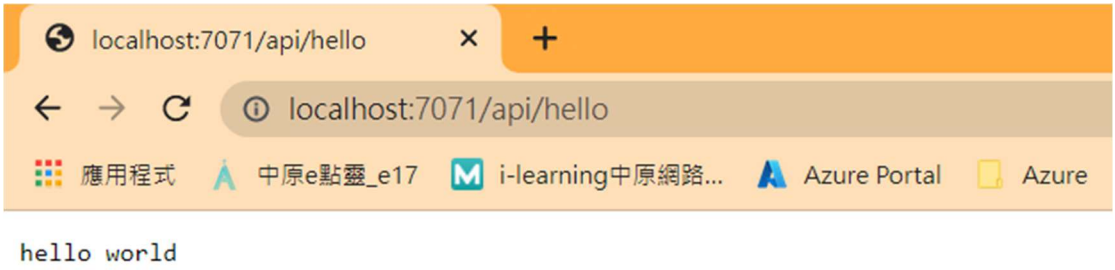
1. Create a function app

	Azure Functions: Create New Project
function project workspace	{Part1}
Select a language	Custom Handler
Select a template for your first function	HttpTrigger
Provide a function name	hello
授權等級	Anonymous



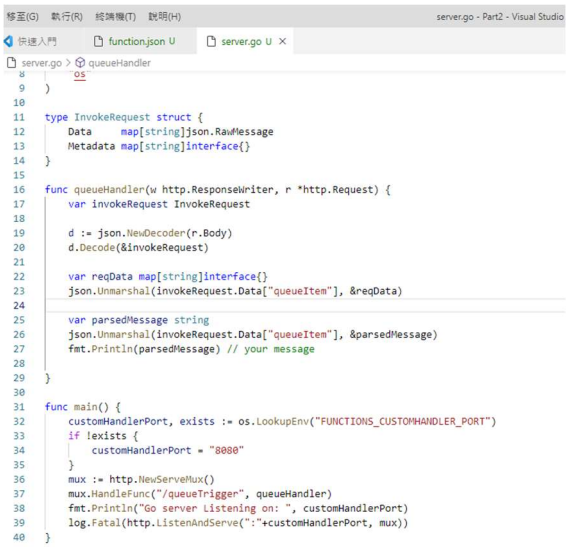
2. config host.json & build execution

3. Run app

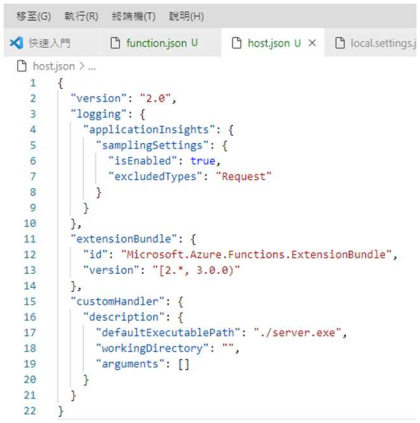


1. Create a function app

	Azure Functions: Create New Project
Select the folder that will contain your function project	{Part2}
Select a language	Custom Handler
Select a template for your first function	HttpTrigger
Provide a function name	queueTrigger
授權等級	Anonymous



2. config host.json & build execution

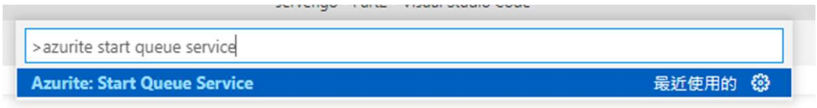


```
{
  "version": "2.0",
  "logging": {
    "applicationInsights": {
      "samplingSettings": {
        "isEnabled": true,
        "excludedTypes": "Request"
      }
    }
  },
  "extensionBundle": {
    "id": "Microsoft.Azure.Functions.ExtensionBundle",
    "version": "[2.*, 3.0.0)"
  },
  "customHandler": {
    "description": {
      "defaultExecutablePath": "./server.exe",
      "workingDirectory": "",
      "arguments": []
    }
  }
}
```

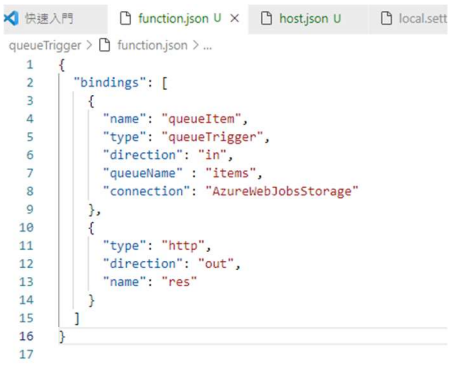
3. config local.settings.json

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage" : "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "custom"
  }
}
```

4. Start Queue Service

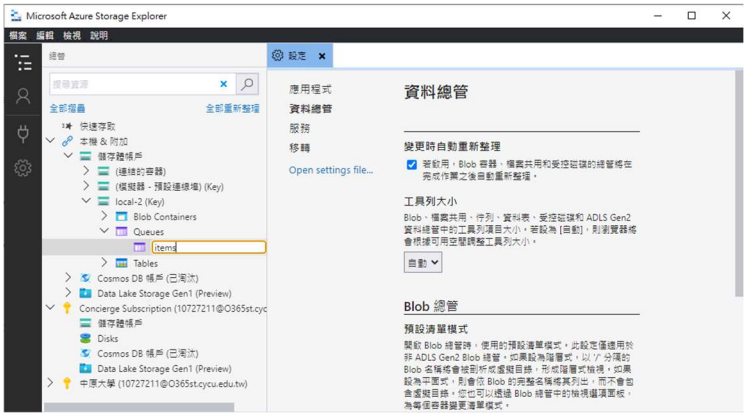


5. config function.json



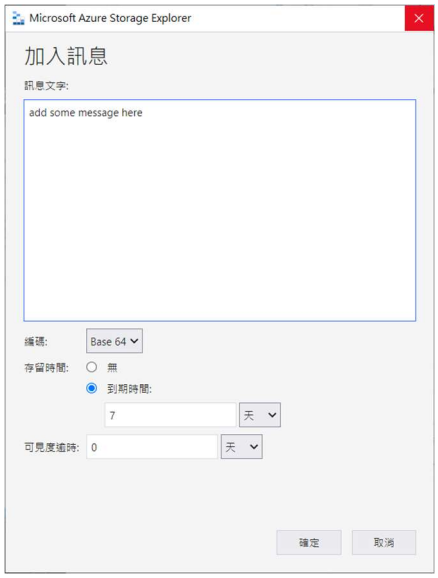
```
{
  "bindings": [
    {
      "name": "queueItem",
      "type": "queueTrigger",
      "direction": "in",
      "queueName" : "items",
      "connection": "AzureWebJobsStorage"
    },
    {
      "type": "http",
      "direction": "out",
      "name": "res"
    }
  ]
}
```

6. Create Queues in Azurite

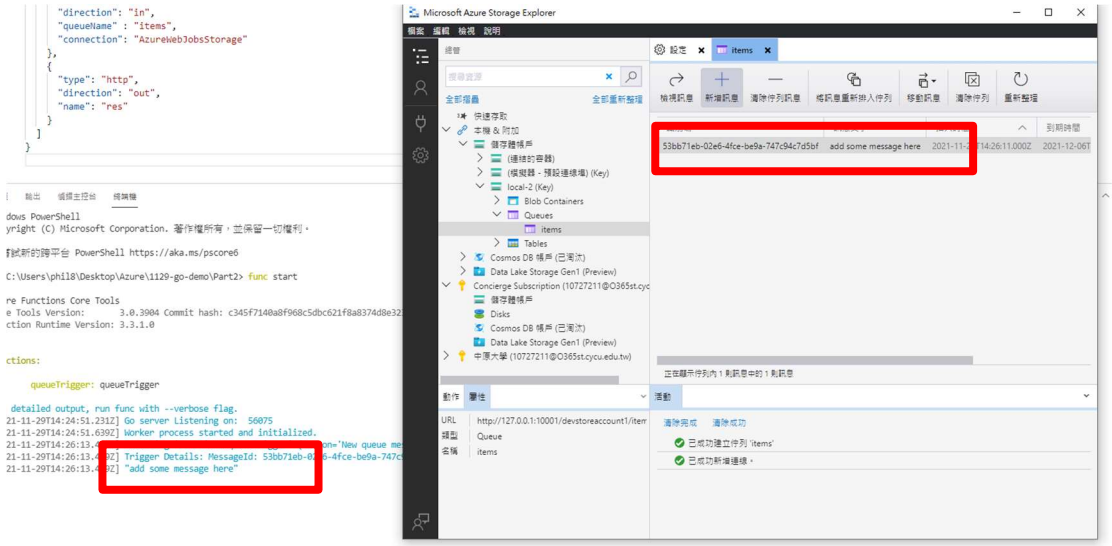


7. Run the app

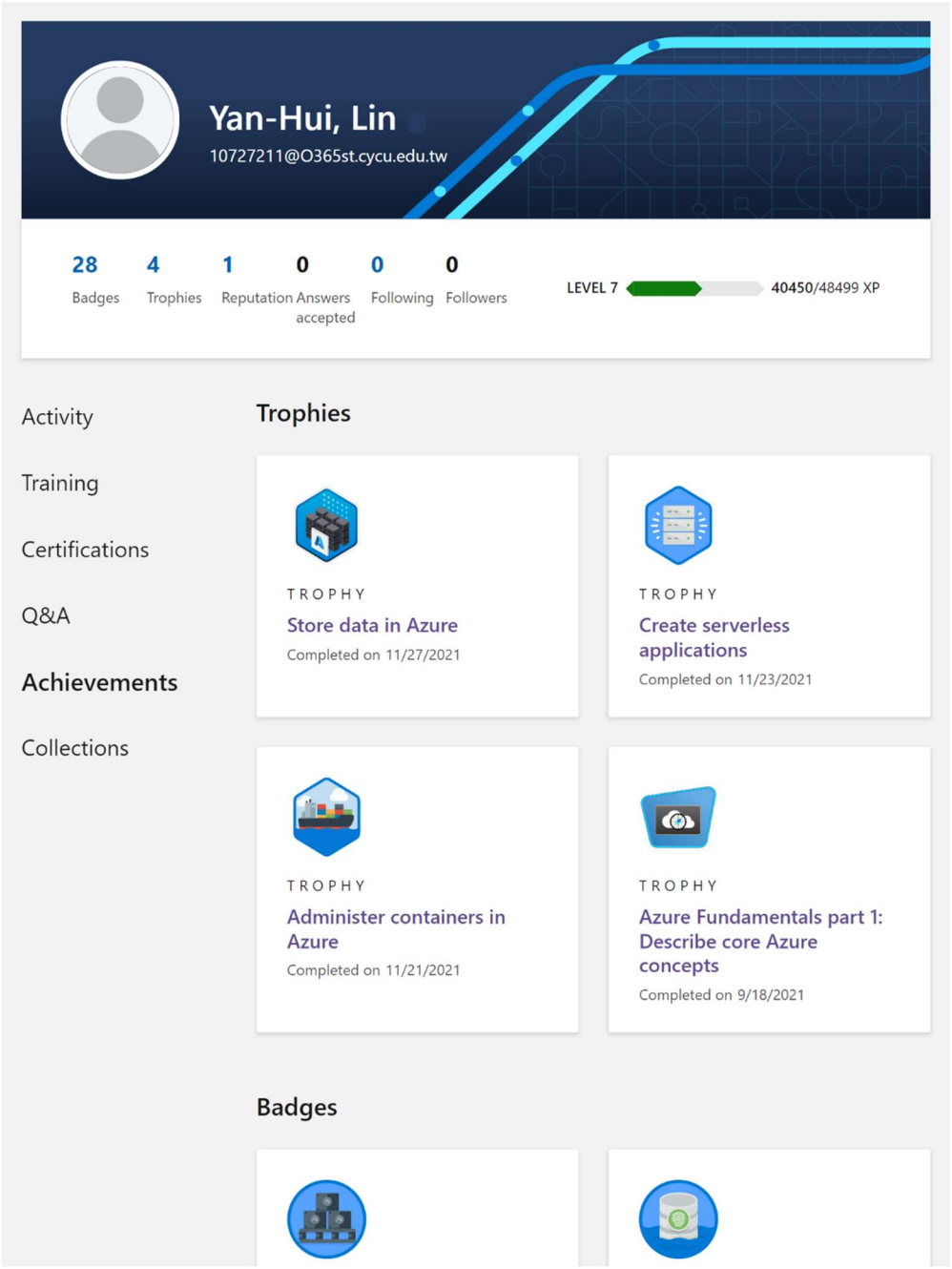
8. Add message in queue



9. Results



Take screenshots of Badges and Trophies



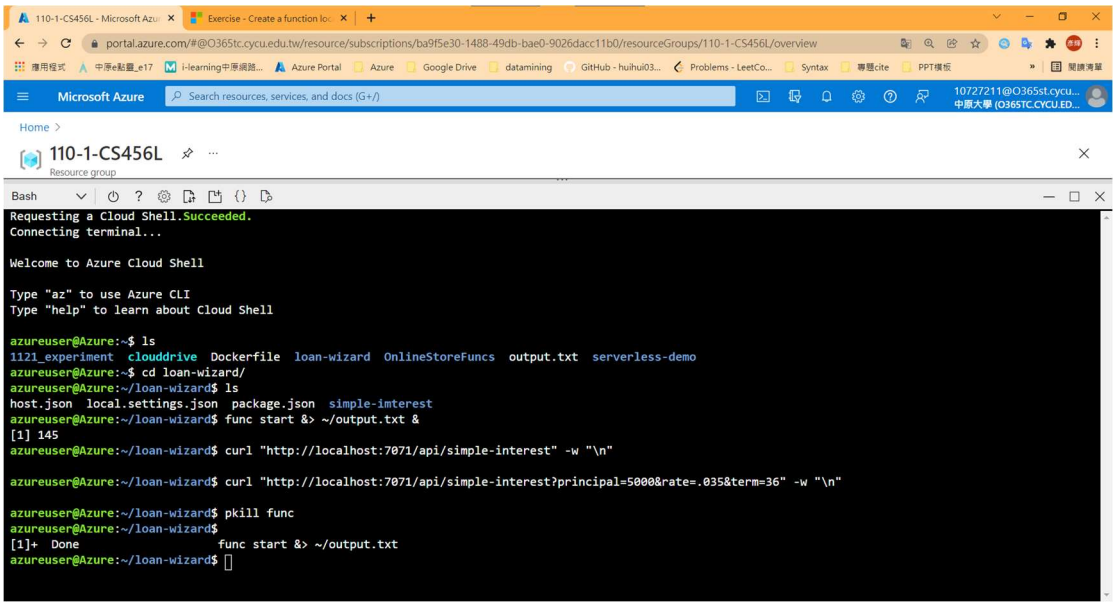
Learned from the Learning Path

在此 Learning Path 中充分實作了 Azure Function 的各種功能，了解 Serverless 的概念，也從 Azure Function 的 Trigger 中看到 Azure 中的其他資源，透過不同的 Trigger 方法搭配不同的 Azure Resource，甚至可以搭配 IDE 做資源的佈署，使得 Azure Function 能實作的功能越來越多，進而看見 Microsoft Azure 的強大生態圈。

Problems

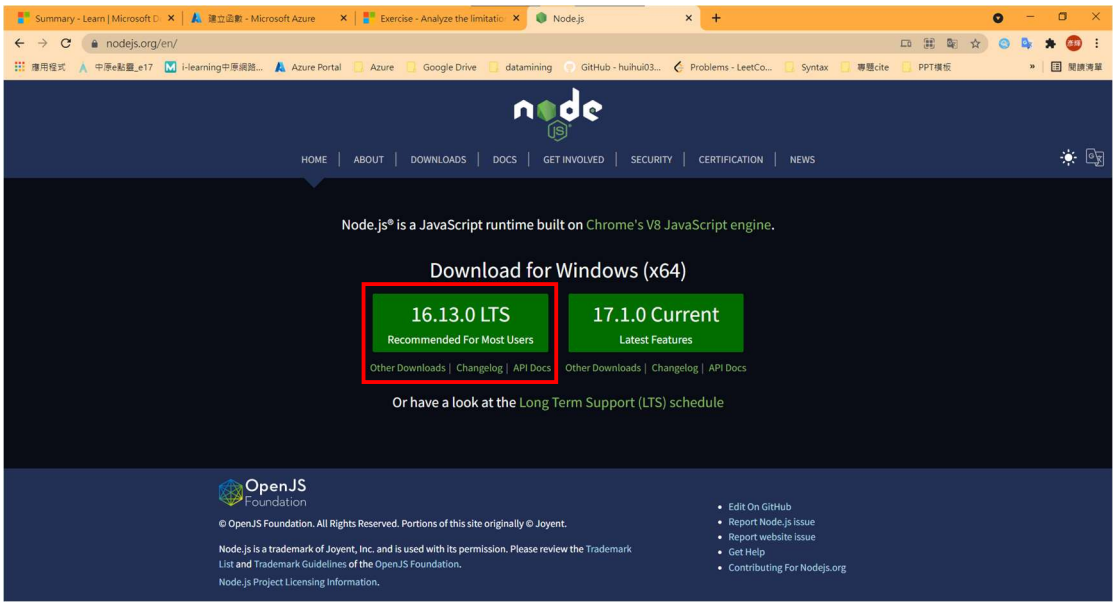
1. Model6

在 Model6 中，使用 Azure Cloud Shell 在背景執行 Azure Functions 時發現錯誤，同樣的方法使用在沙盒中卻能正常運行。

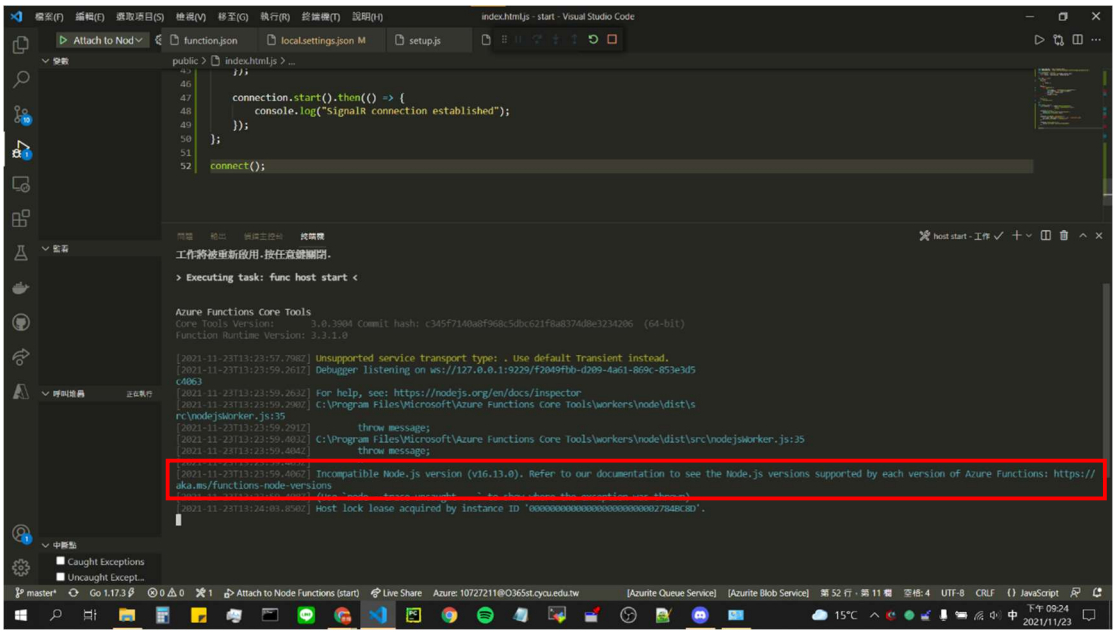


2. Model9

需要下載 Node.js 以接續教程。但官方給的 Node.js 載點中 https://nodejs.org/en/ 會安裝到最新版本的 Node.js、npm（如下圖為 16.13LTS）



但此版本將無法與 Azure Core Functions 相容，在 Start Local Function 將出現 Error。



依序網址來到相容性說明網頁

https://docs.microsoft.com/zh-tw/azure/azure-functions/functions-reference-node?tabs=v2#node-version

節點版本

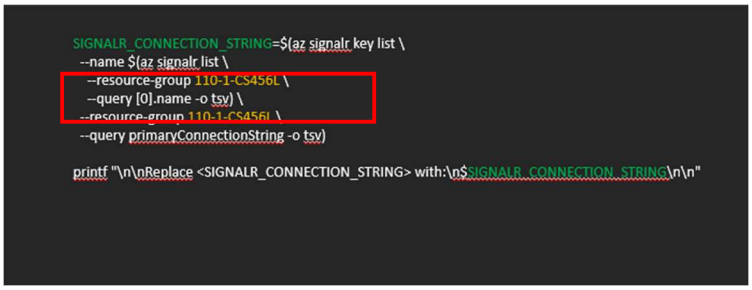
下錶針對每個主要版本的函式執行時間（依作業系統），顯示目前支援的 Node.js 版本：

Functions 版本	節點版本 (Windows)	Linux) (節點版本
4.x	~14	node|14
3. x (建議的)	~14 (建議的) ~12 ~10	node|14 (建議的) node|12 node|10
2.x	~12 ~10 ~8	node|10 node|8
1.x	6.11.2 (由執行階段鎖定)	n/a

將 Node.js 降版後即可解決問題(v14) https://nodejs.org/download/release/v14.18.1/

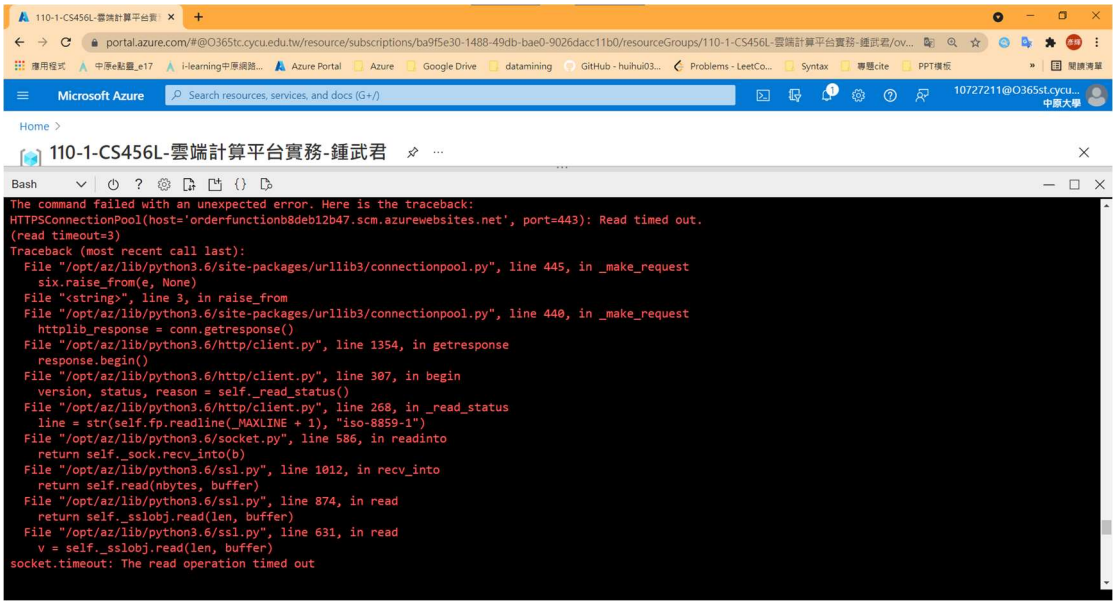
3. Model 9

在拿 SingalR key String 時，範例程式碼中的 query[0]有可能導致 Error（將 index 寫死，將拿到第一個 SignalR Resource 的 key，但「第一個 SignalR Resource」並非剛創立的 Resource）



4. Model 10

在 110-1-456L 使用官方提供之腳本時出現 Error，原因不明，環境更換為沙盒時即可恢復運作。



5. Model11 – Unit4

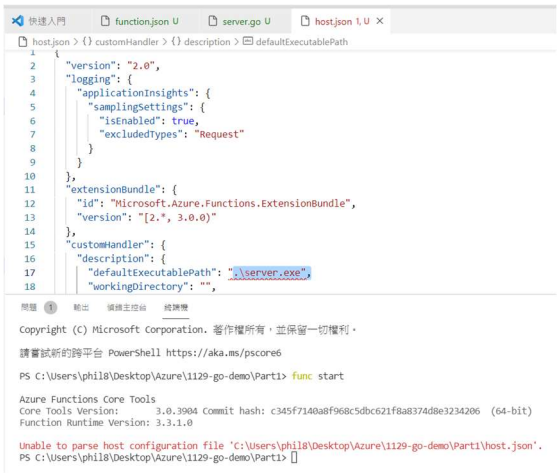
在 Windows 環境下，路徑使用'\'將 Error，應改為'/'。

1. From a terminal, run `go build server.go` in the project root:

```
Go
go build server.go
```

This step creates an executable file that's called *server* on macOS and Linux, or *server.exe* on a Windows OS.

2. Open the *host.json* file and find the `defaultExecutablePath` element inside the `customHandler` one. Specify `./server` on macOS and Linux, or `.\server.exe` on a Windows OS.

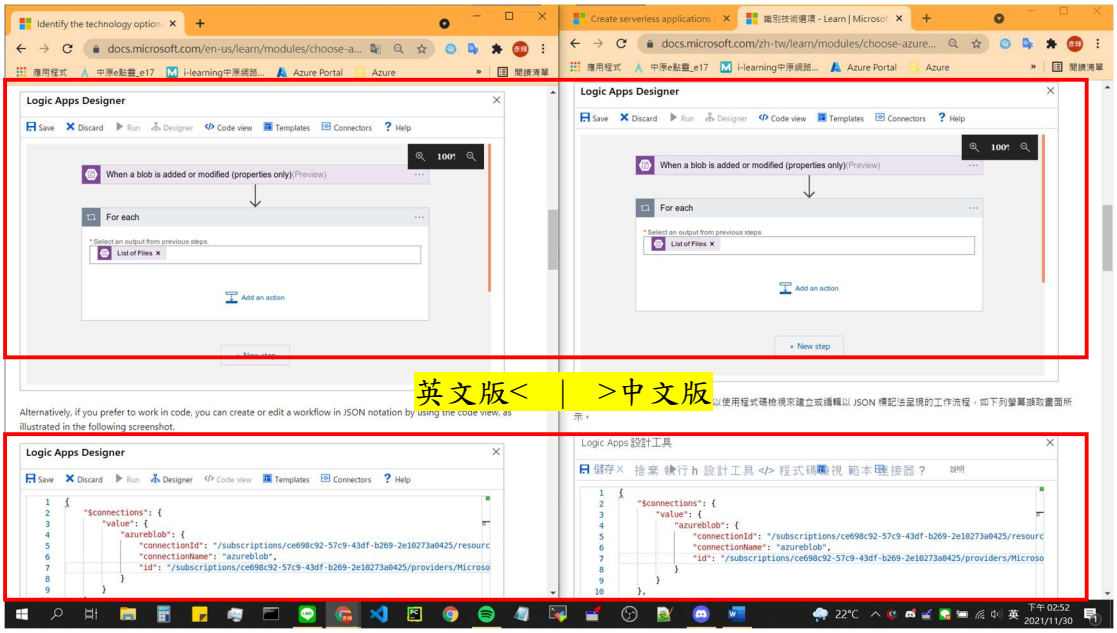


FeedBack

中文翻譯改進

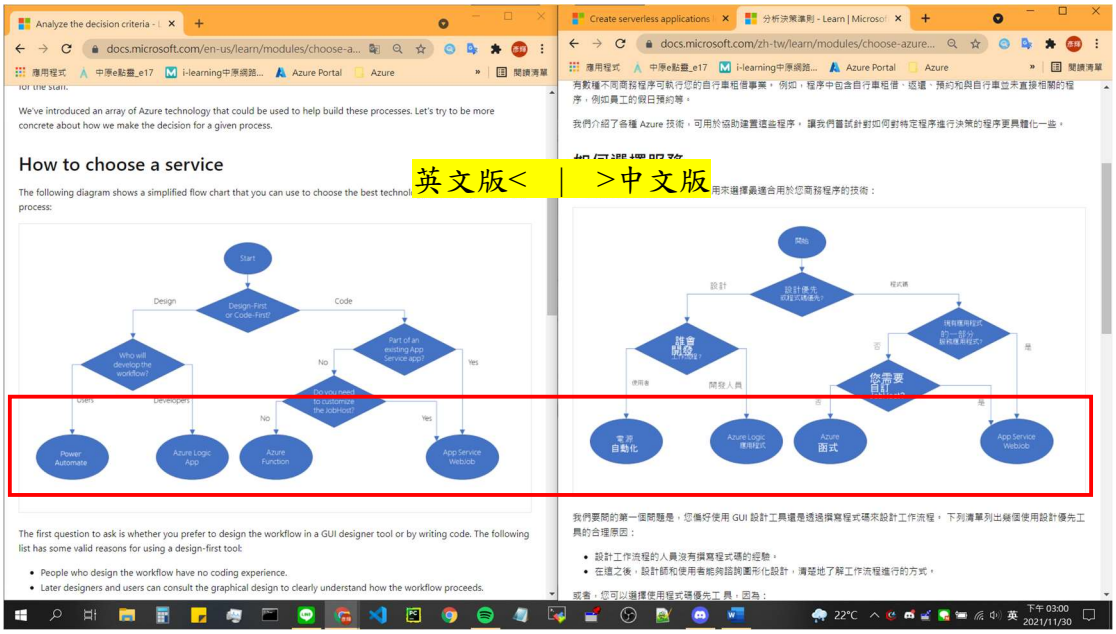
1. Model1 – Unit2

左邊為英文版，右邊為中文版，過程中看到教程會穿插圖片做解說，但解說圖片的文字翻譯程度卻不統一，翻譯過後的可讀性甚至更低。



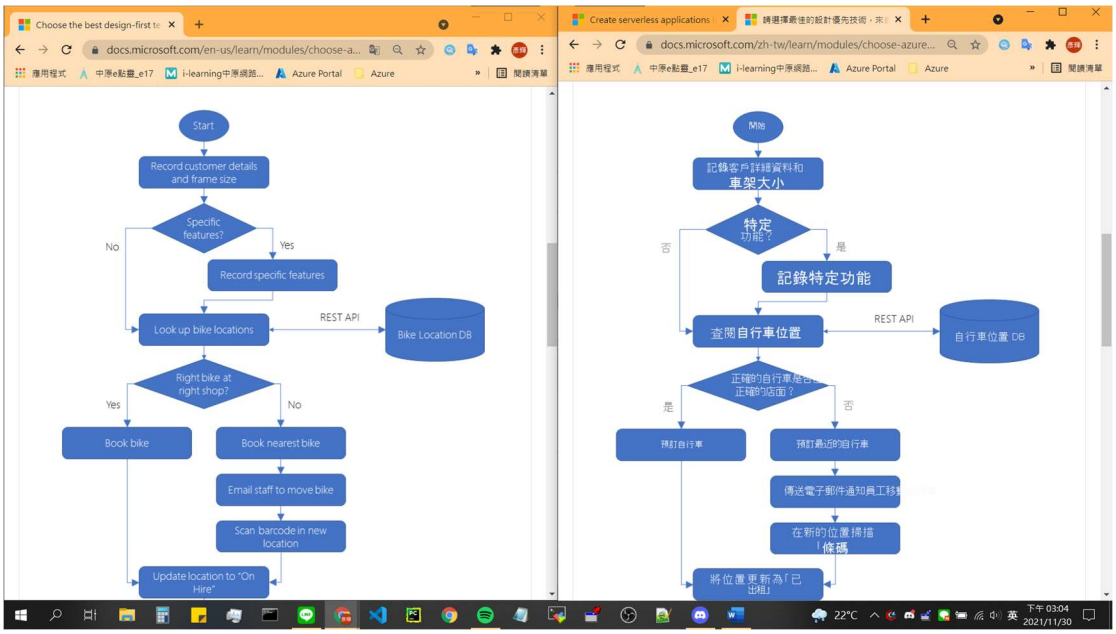
2. Model1 – Unit3

產品名稱經過翻譯可讀性降低。



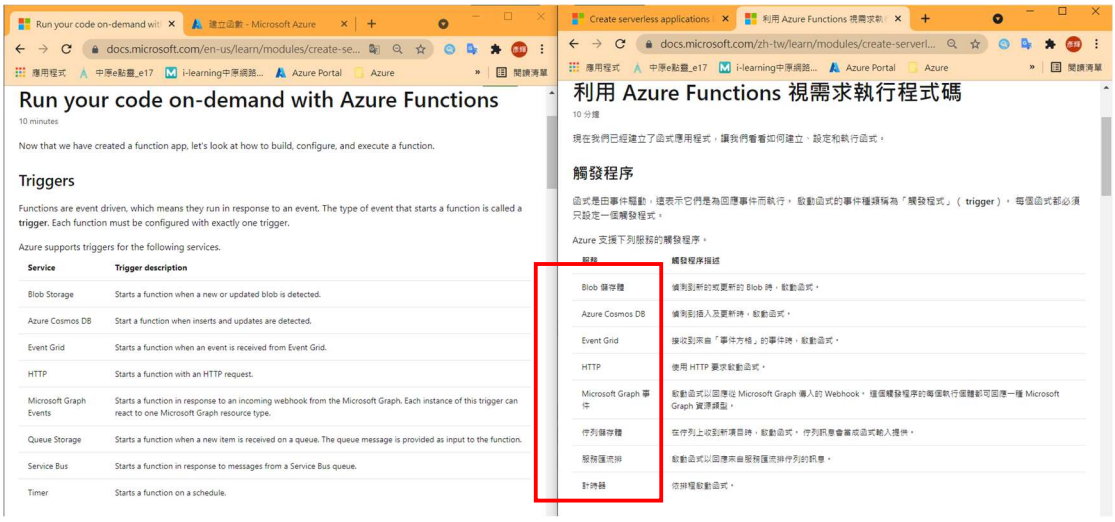
3. Model1 – Unit4

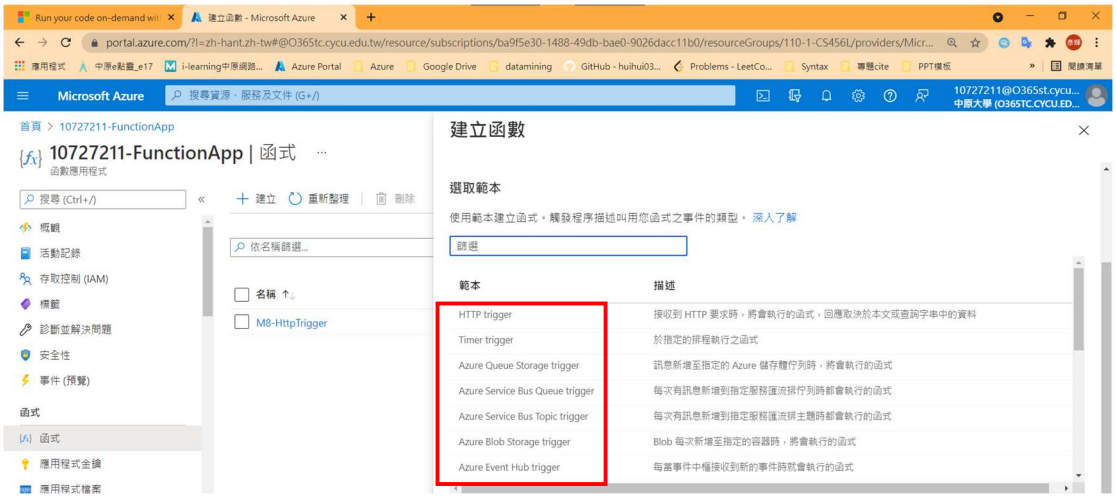
排版。



4. Model2 – Unit4

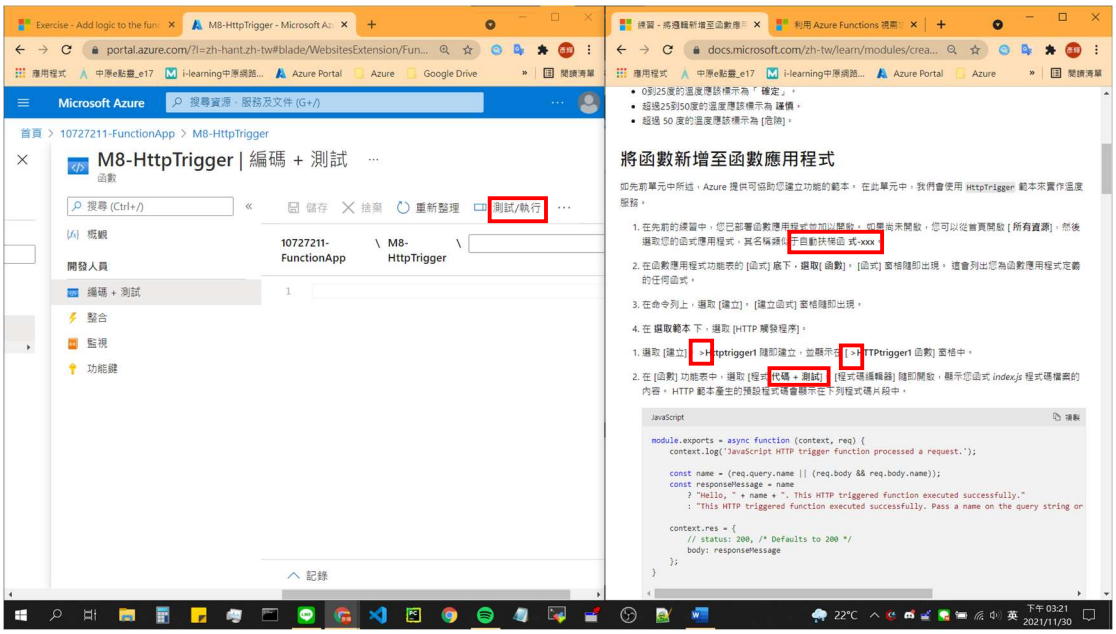
服務名稱要統一英文版本，在 Azure Portal 上創建名稱時，服務為英文版本。





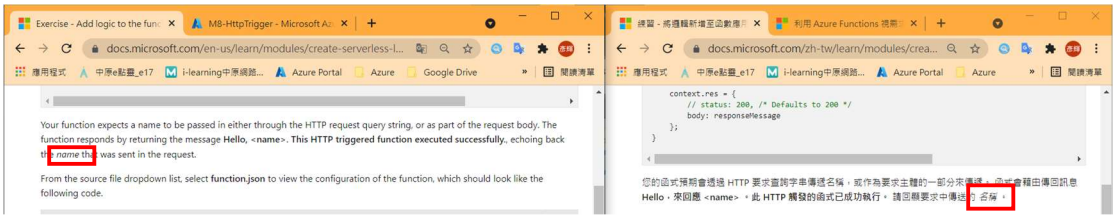
5. Model2 – Unit5

名稱不統一。



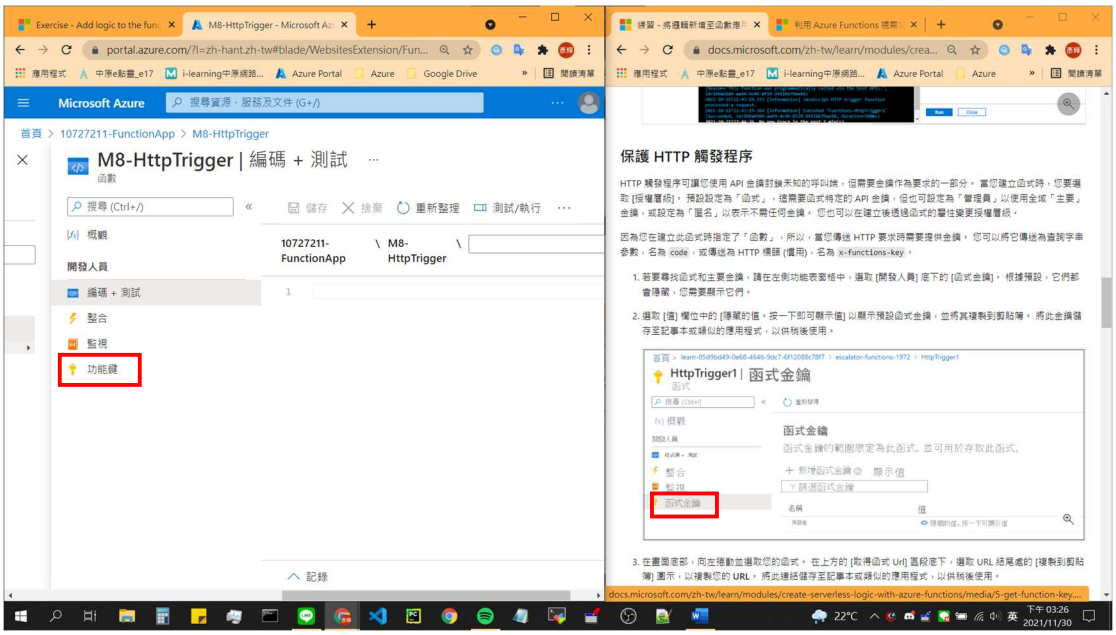
6. Model2 – Unit5

變數名稱被翻譯。



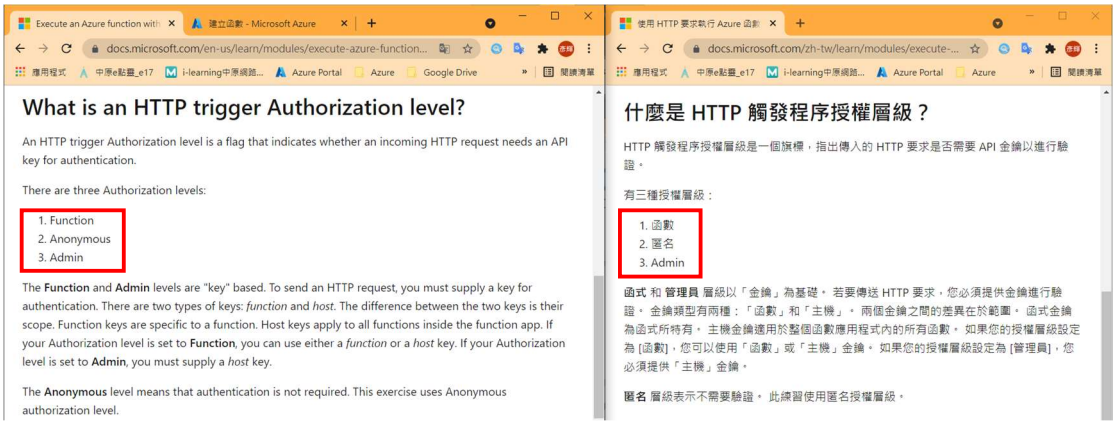
7. Model2 – Unit5

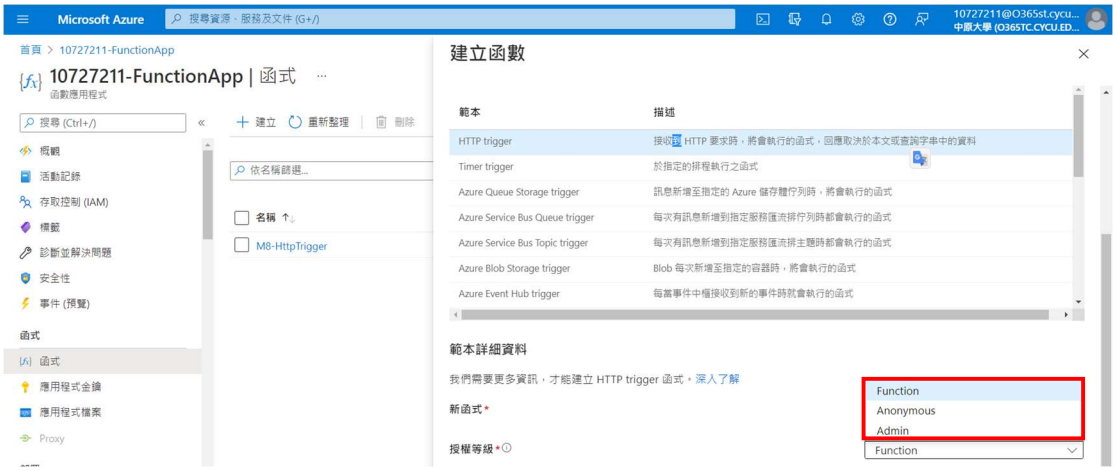
名稱不統一。



8. Model3 – Unit2

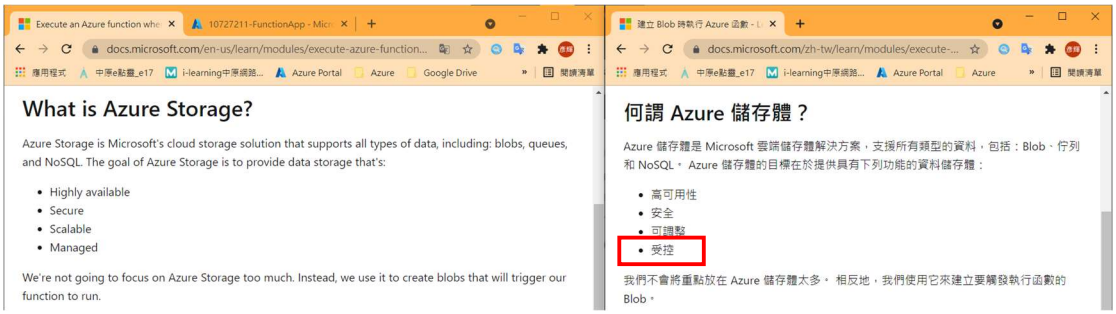
服務名稱要統一英文版本，在 Azure Portal 上創建名稱時，服務為英文版本。



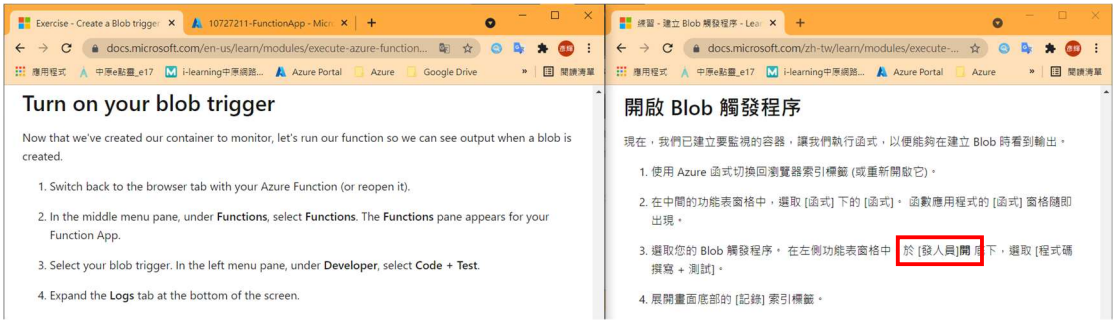


9. Model 3 – Unit7

「管理」也許是個更好的翻譯。

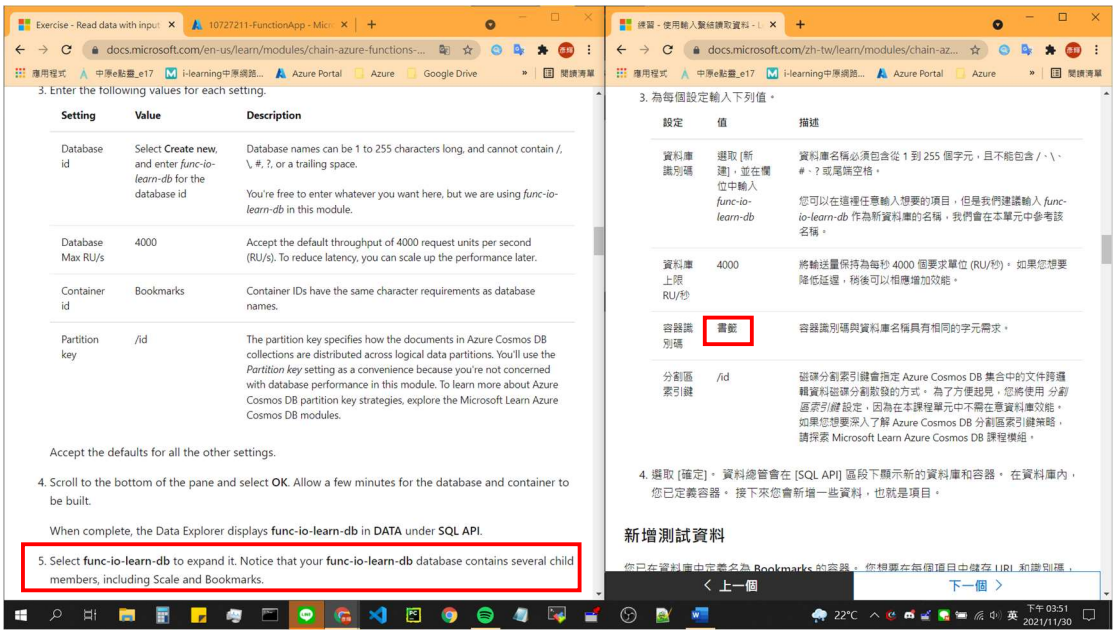


10. Model3 – Unit8



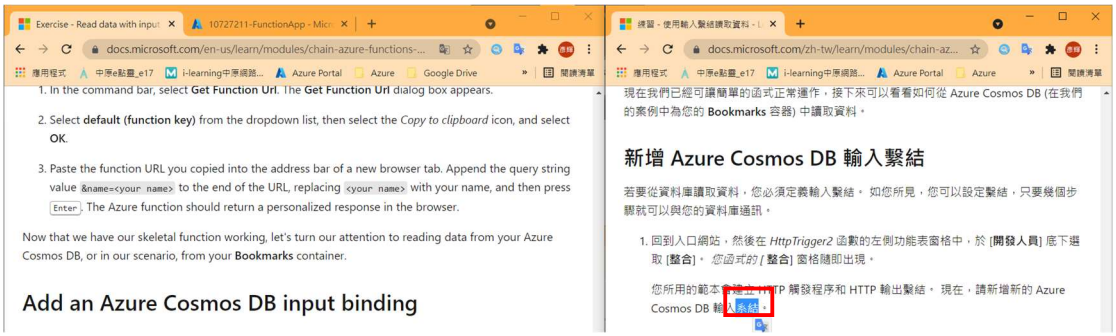
11. Model4 – Unit5

變數名稱不能被翻譯&有部分內容未被翻譯



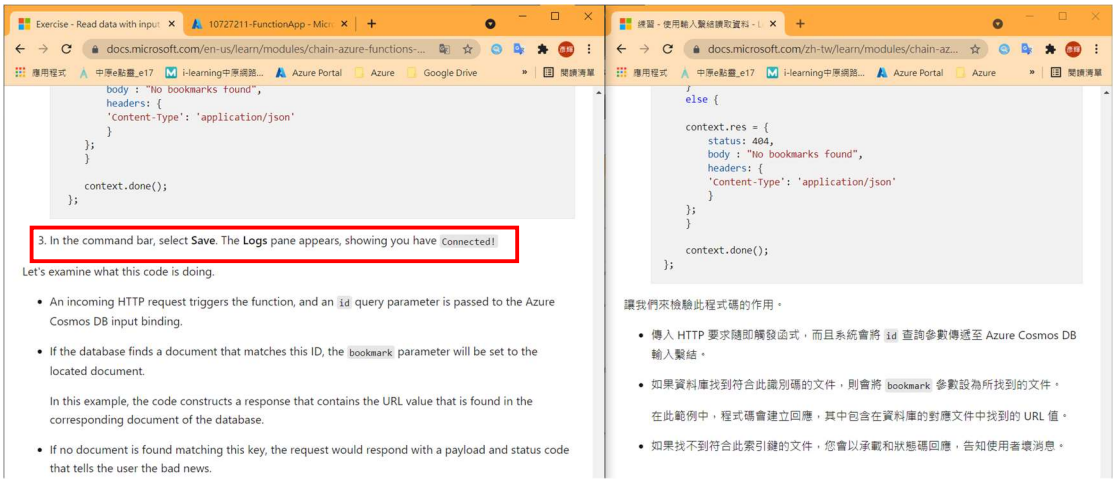
12. Model4 – Unit5

翻譯詞要統一。



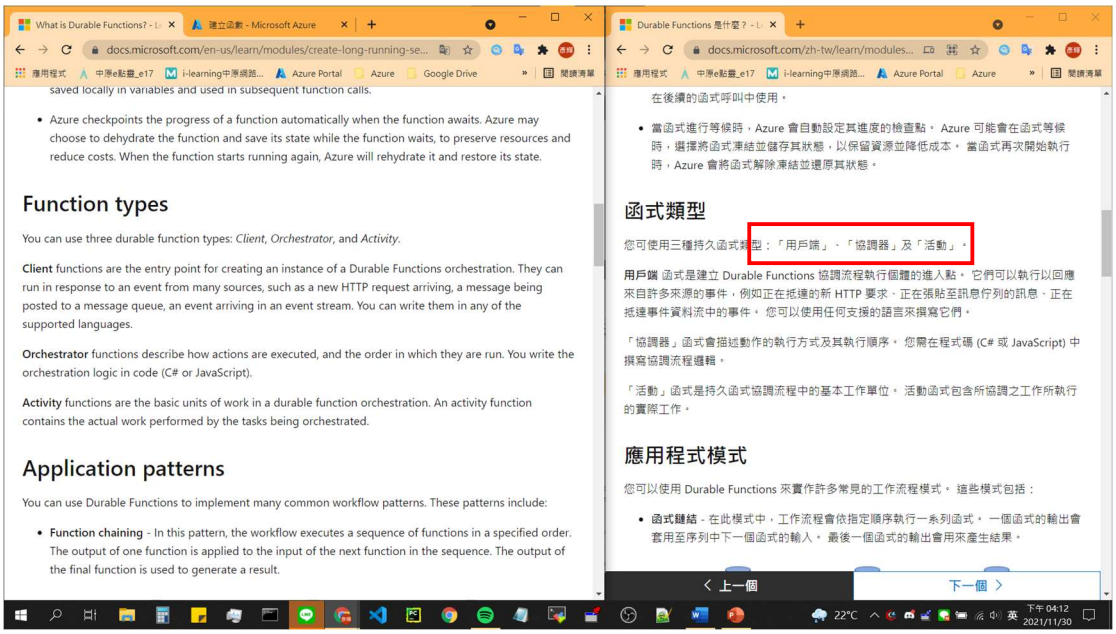
13. Model4 – Unit5

有部分內容未被翻譯。



14. Model5 – Unit2

服務名稱要統一英文版本，在 Azure Portal 上創建名稱時，服務為英文版本。



Function types

You can use three durable function types: *Client*, *Orchestrator*, and *Activity*.

Client functions are the entry point for creating an instance of a Durable Functions orchestration. They can run in response to an event from many sources, such as a new HTTP request arriving, a message being posted to a message queue, an event arriving in an event stream. You can write them in any of the supported languages.

Orchestrator functions describe how actions are executed, and the order in which they are run. You write the orchestration logic in code (C# or JavaScript).

Activity functions are the basic units of work in a durable function orchestration. An activity function contains the actual work performed by the tasks being orchestrated.

Application patterns

You can use Durable Functions to implement many common workflow patterns. These patterns include:

- **Function chaining** - In this pattern, the workflow executes a sequence of functions in a specified order. The output of one function is applied to the input of the next function in the sequence. The output of the final function is used to generate a result.

函式類型

您可使用三種持久函式類型：「用戶端」、「協調器」及「活動」。

用戶端 函式是建立 Durable Functions 協調流程執行個體的進入點。 它們可以執行以回應來自許多來源的事件，例如正在抵達的新 HTTP 要求、正在張貼至訊息佇列的訊息、正在抵達事件資料流中的事件。 您可以使用任何支援的語言來撰寫它們。

「協調器」函式會描述動作的執行方式及其執行順序。 您需在程式碼 (C# 或 JavaScript) 中撰寫協調流程邏輯。

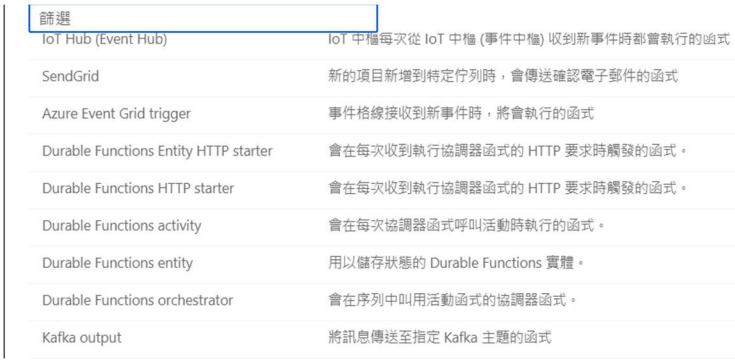
「活動」函式是持久函式協調流程中的基本工作單位。 活動函式包含所協調之工作所執行的實際工作。

應用程式模式

您可以使用 Durable Functions 來實作許多常見的工作流程模式。 這些模式包括：

- **函式鏈結** - 在此模式中，工作流程會依指定順序執行一系列函式。 一個函式的輸出會套用至序列中下一個函式的輸入。 最後一個函式的輸出會用來產生結果。

< 上一個　　下一個 >



篩選

IoT Hub (Event Hub)	IoT 中樞每次從 IoT 中樞 (事件中樞) 收到新事件時都會執行的函式
SendGrid	新的項目新增到特定佇列時，會傳送確認電子郵件的函式
Azure Event Grid trigger	事件格線接收到新事件時，將會執行的函式
Durable Functions Entity HTTP starter	會在每次收到執行協調器函式的 HTTP 要求時觸發的函式。
Durable Functions HTTP starter	會在每次收到執行協調器函式的 HTTP 要求時觸發的函式。
Durable Functions activity	會在每次協調器函式呼叫活動時執行的函式。
Durable Functions entity	用以儲存狀態的 Durable Functions 實體。
Durable Functions orchestrator	會在序列中叫用活動函式的協調器函式。
Kafka output	將訊息傳送至指定 Kafka 主題的函式

15. Model6 – Unit2

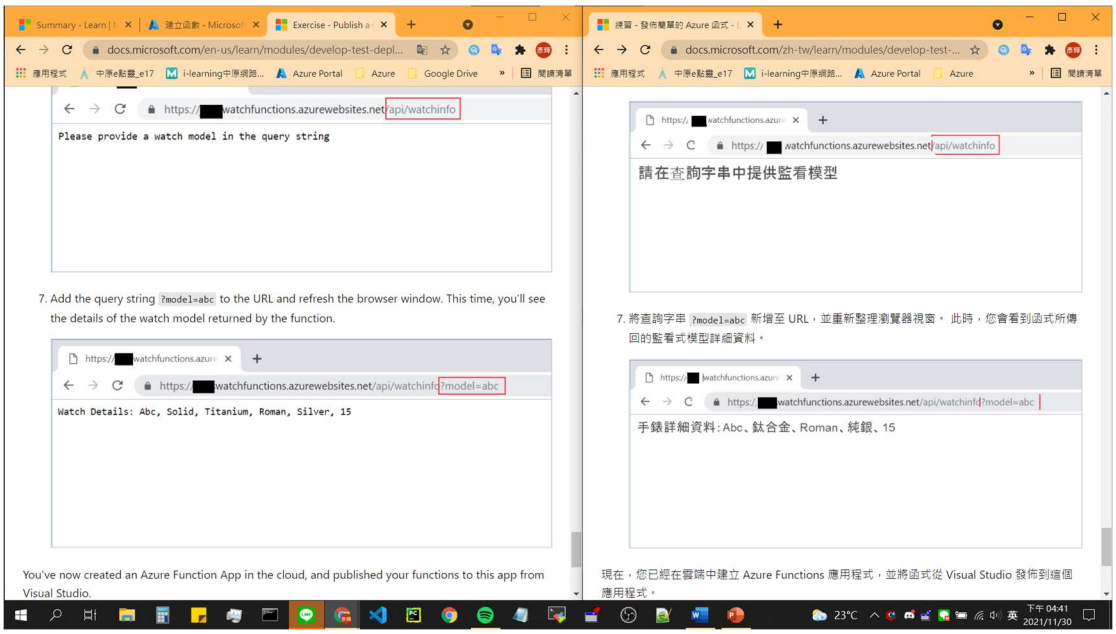
log 可以不必翻譯

```
user@Azure:~$ func
Azure Functions Core Tools (2.7.1724 Commit hash: faf97bb64a53fbf1c7ef7e5c608c392c8662951b)
Function Runtime Version: 2.0.12763.0
Usage: func [context] [context] <action> [-/--options]

Contexts:
azure       Commands to log in to Azure and manage resources
durable     Commands for working with Durable Functions
extensions  Commands for installing extensions
function    Commands for creating and running functions locally
host        Commands for running the Functions host locally
kubernetes  Commands for working with Kubernetes and Azure Functions
settings    Commands for managing environment settings for the local Functions host
templates   Commands for listing available function templates

Actions:
start   Launches the functions runtime host
    --port [-p]         Local port to listen on. Default: 7071
    --cors              A comma separated list of CORS origins with no spaces. Example: https://functions.azure.com
                        https://functions-staging.azure.com
```

Core Tools 是獨立的公用程式，而不是大型整合式開發環境或工具的延伸模組。 您只需使用命令列和 Core Tools 即可完成上述工作，而您也可以使用您想要的任何文字編輯器來撰寫程式碼和修改設定。 在實務上，您也需要 Azure CLI 或 Azure PowerShell 來登入 Azure 並執行其他管理工作。

16. Model7 – Unit5

過度翻譯。



17. Model10 – Unit3

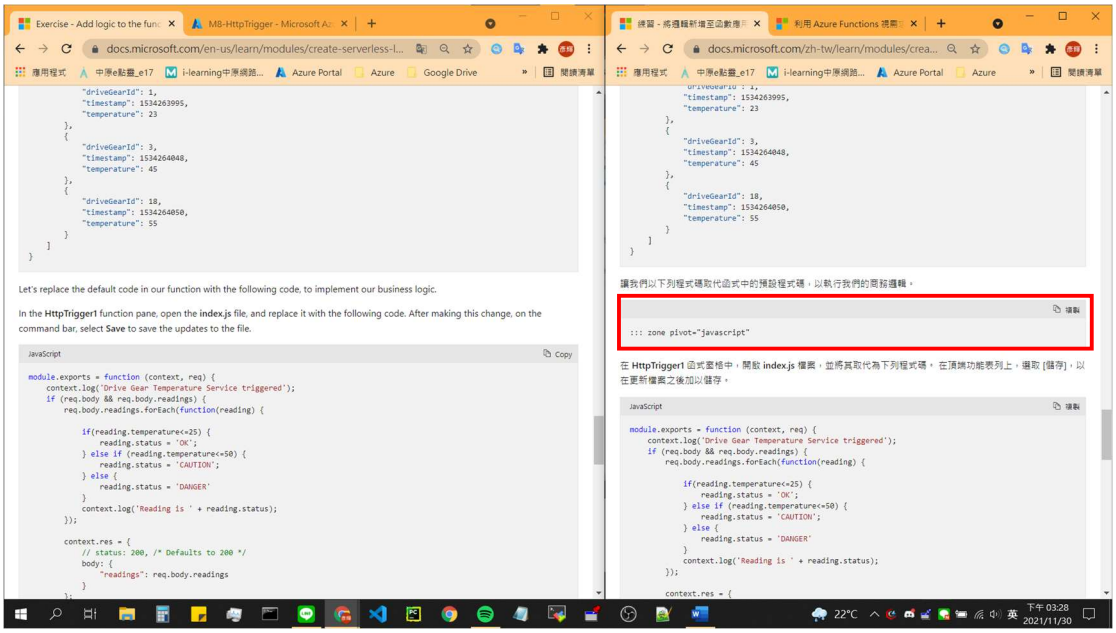
變數名稱不應被翻譯。

18. Model10 – Unit3

變數名稱不應被翻譯。

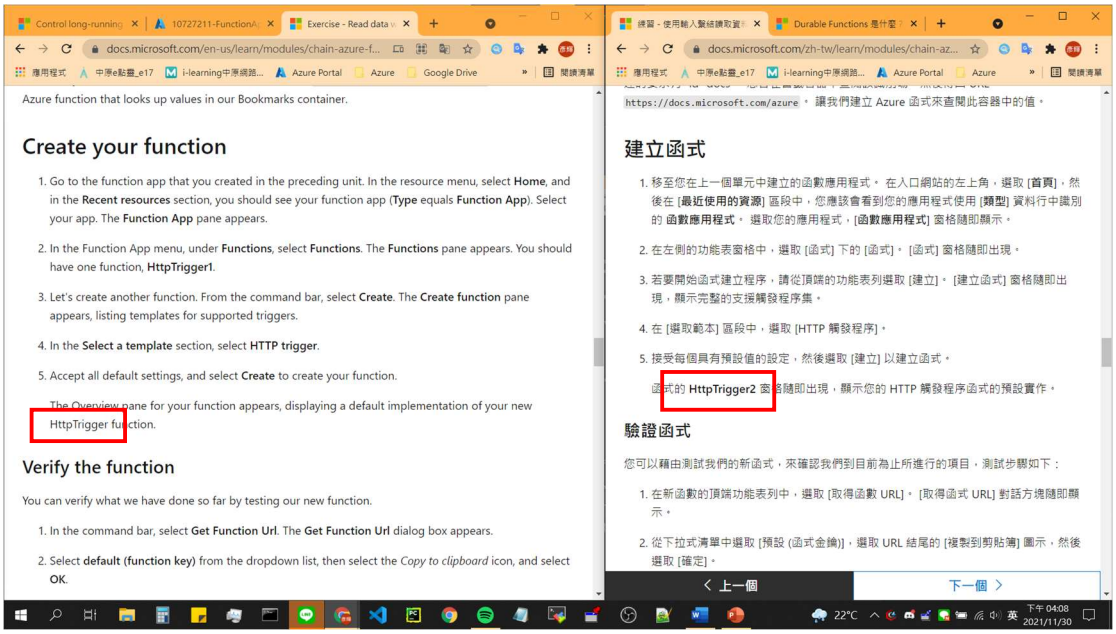
網頁 Bug

1. Model2 – Unit5



2. Model4 – Unit5

在 Unit3 創造的 HttpTrigger 並沒有接續著使用，在 Unit4 又創造一個 HttpTrigger & 翻譯錯誤

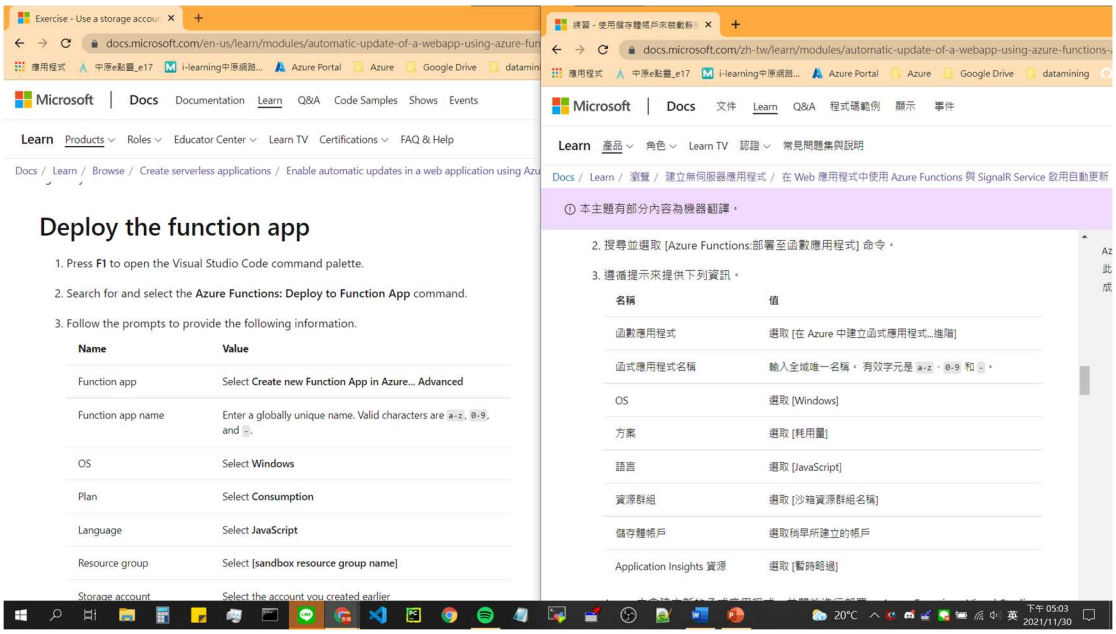


3. Model7 – Unit5

格式錯誤

4. Model9 – Unit7

應更正為以下格式



	Azure Functions: Deploy to Function App
Template	Create new Function App in Azure... Advanced
Name	10727211-deployfunctionapp
Running Stack	Node.js 8 LTS
Resource Group	110-1-CS456L
Location for new resource	東南亞 (for CS456L)
Hosting Plan	Consumption
storage	10727211model9storage
Application Insights resource	Skip for now