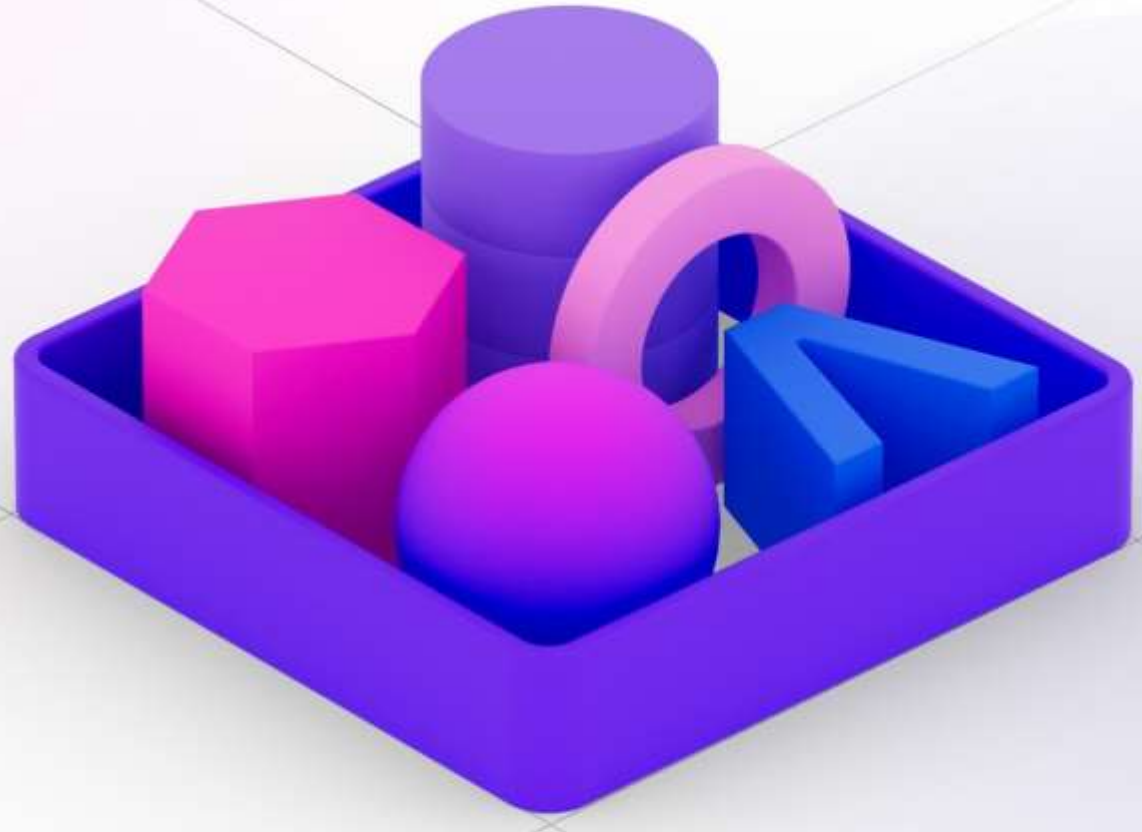
STUDY4
為學習而生

Build School

.NET Conf
TAIWAN



STUDY4 為學習而生 Build School

Deploy Azure Function with KEDA on AKS

Roberson Liou

About

- Backend / DevOps / Cloud
- Microsoft MVP(2020-2024)
- twMVC / DevOps Taiwan 志工
- Blog - 工程良田的小球場

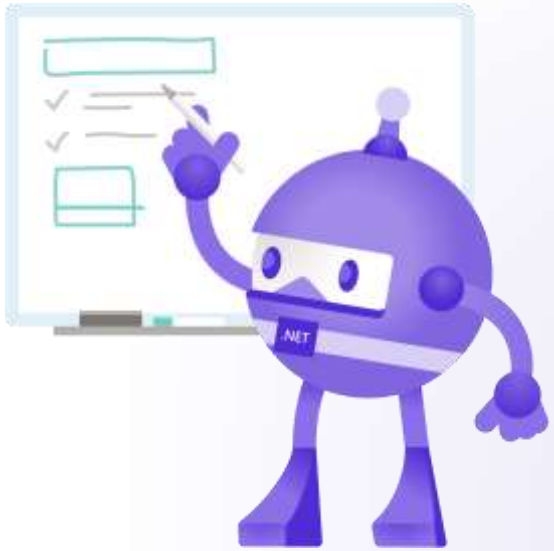
twMVC

DevOps Taiwan

Outline

- Introduction
- Hello KEDA Function
- Deployment Workflow
- Other Tips

Introduction



What is Azure Function

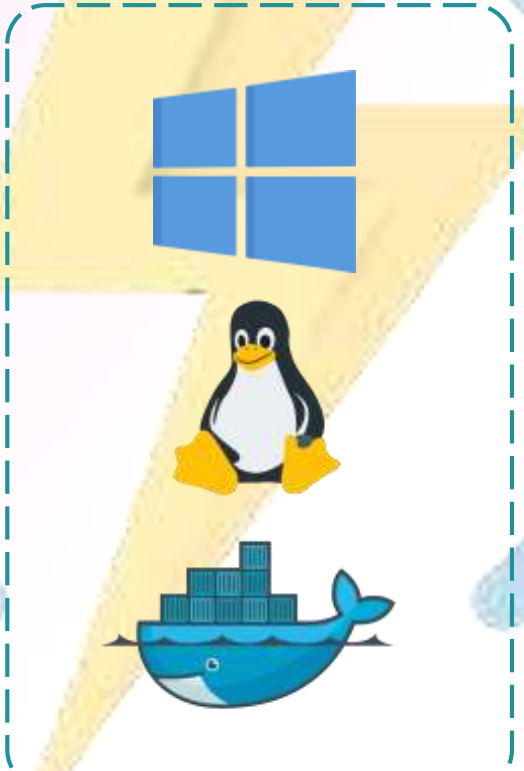
- Serverless 雲端運算服務
- 可依需求自動擴展機器數量
- 可使用多種語言進行開發
- 支援跨平台運行

Azure Function Ecosystem

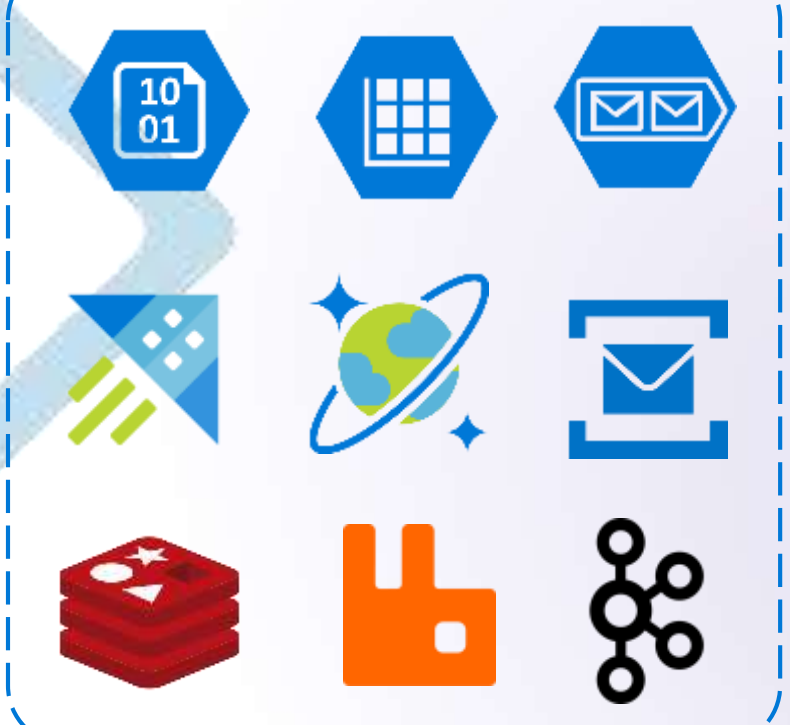
Languages

Supported languages

Hosting



Triggers



Triggers and bindings

Language Support Platforms

Language	Runtime stack	Linux	Windows	In-portal editing
C# (isolated worker model)	.NET	✓	✓	
C# (in-process model)	.NET	✓	✓	
C# script	.NET	✓	✓	✓
JavaScript	Node.js	✓	✓	✓
Python	Python	✓		✓
Java	Java	✓	✓	
PowerShell	PowerShell Core	✓	✓	✓
TypeScript	Node.js	✓	✓	
Go/Rust/other	Custom Handlers	✓	✓	

Process Model(1)

- 分為 Isolated process 及 In-process
- 兩者所使用的 Function SDK 不同
  - 要安裝的 Nuget package 不同
- Isolated process 不須依賴於 host process
  - 同個 host 可同時執行不同版本的 function
- In-process 的版本須與其 host 版本相關

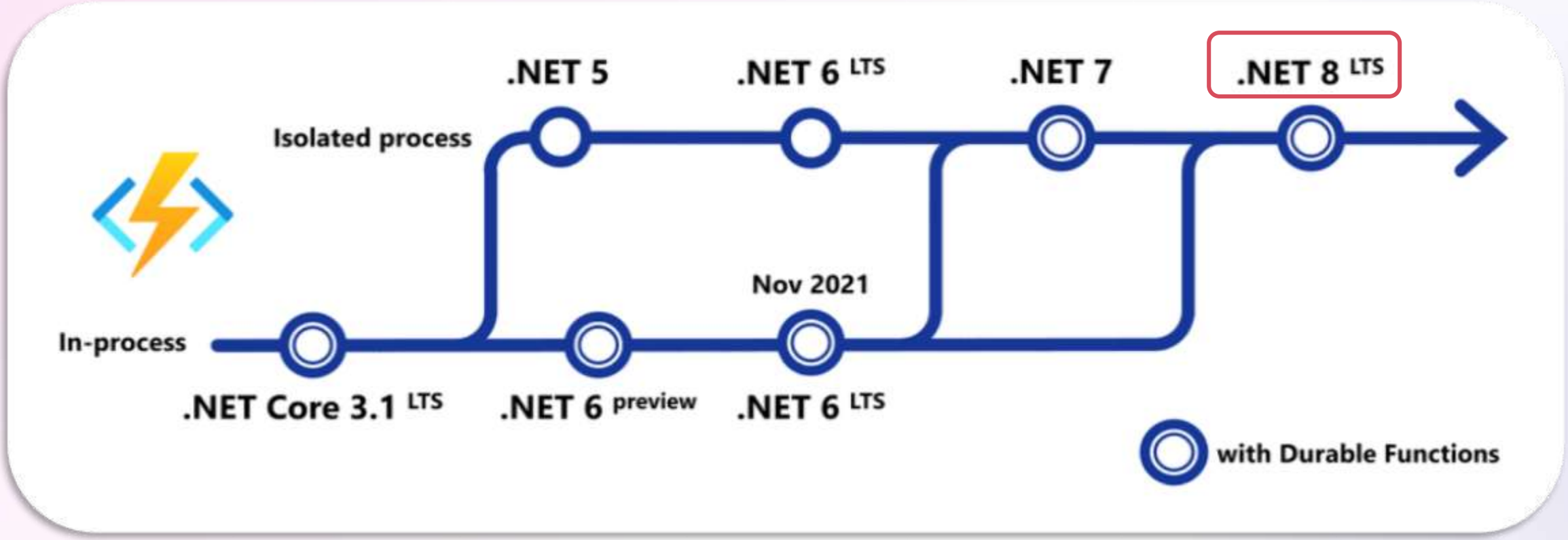
Process model - Isolated vs. In-process

Process Model(2)

- 強力推薦使用 Isolated process
- 未來 In-process 僅支援 LTS 版本
- Container Image Size 差異
  - Isolated: 572MB (.NET 8)
    - azure-functions/dotnet-isolated:4-dotnet-isolated8.0
  - In-process: 1.18GB (.NET 6)
    - Azure-functions/dotnet:4

Process model - Isolated vs. In-process

Process Model(3)

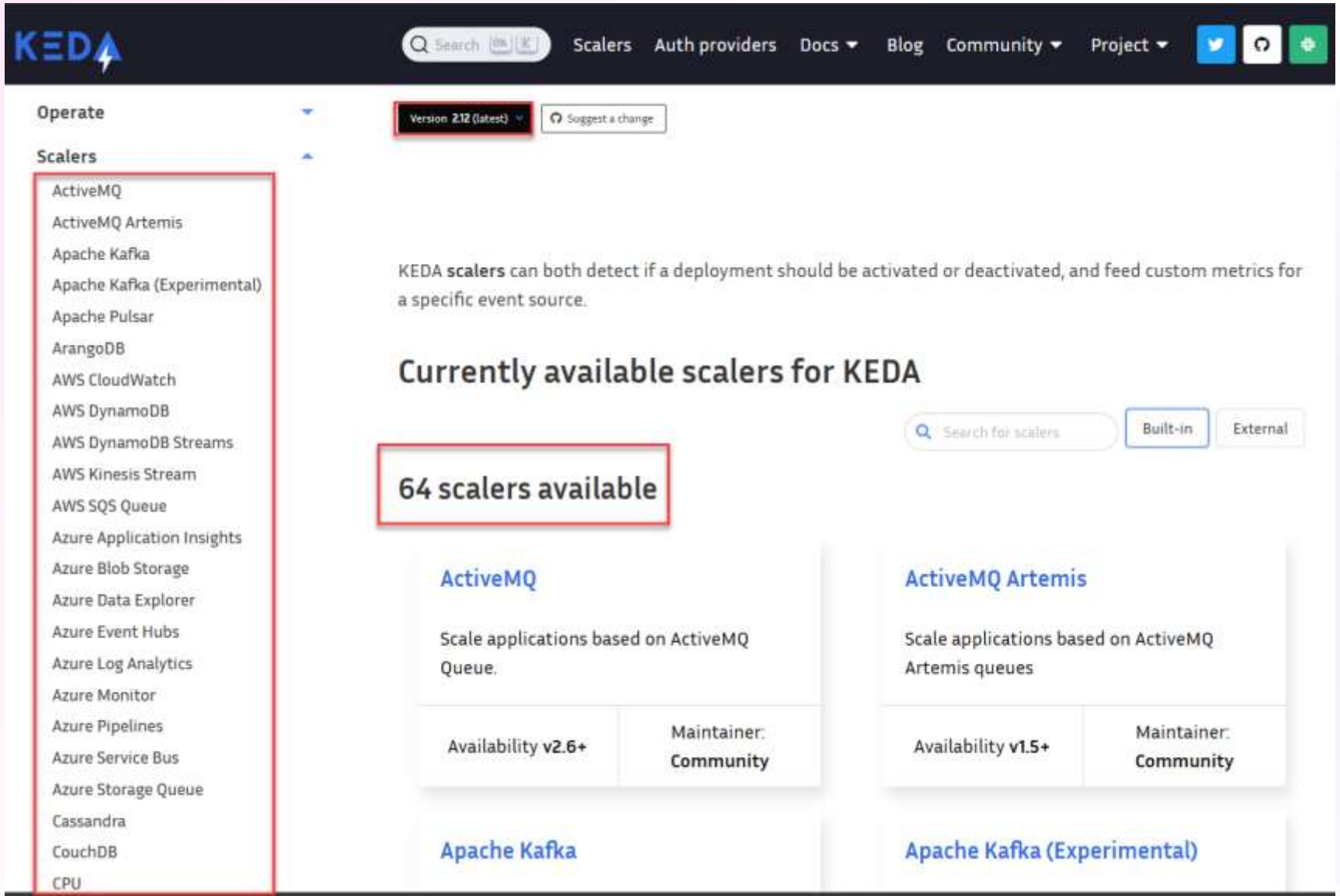


Process model - Isolated vs. In-process

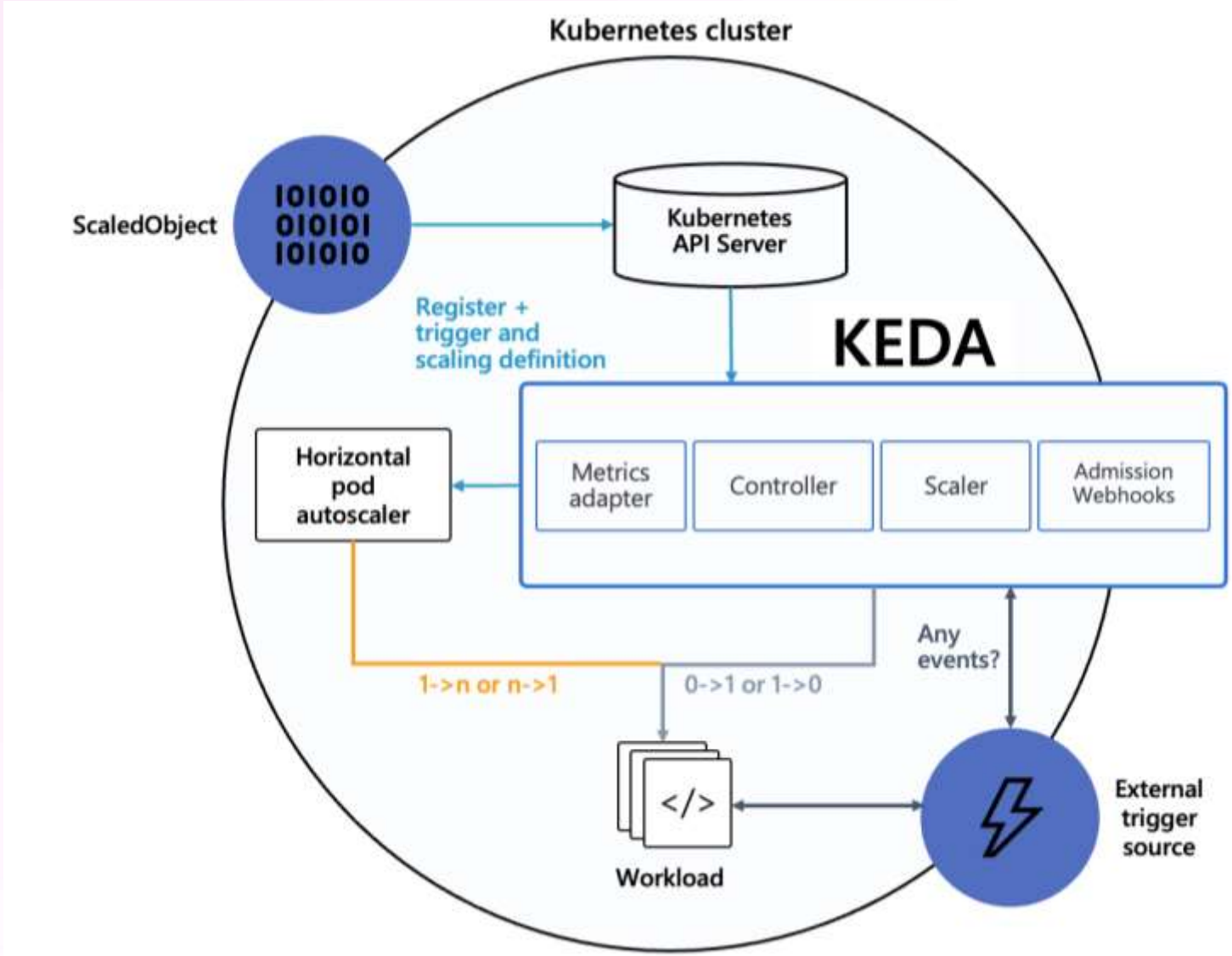
What is KEDA

- Kubernetes Event-Driven Autoscaling
- 今年 8 月剛從 CNCF 畢業
- 目前支援 60+ 事件來源
- 可動態擴展及縮放 instance 數量
  - from 0 to N

KEDA Scalers – v2.12



How KEDA works

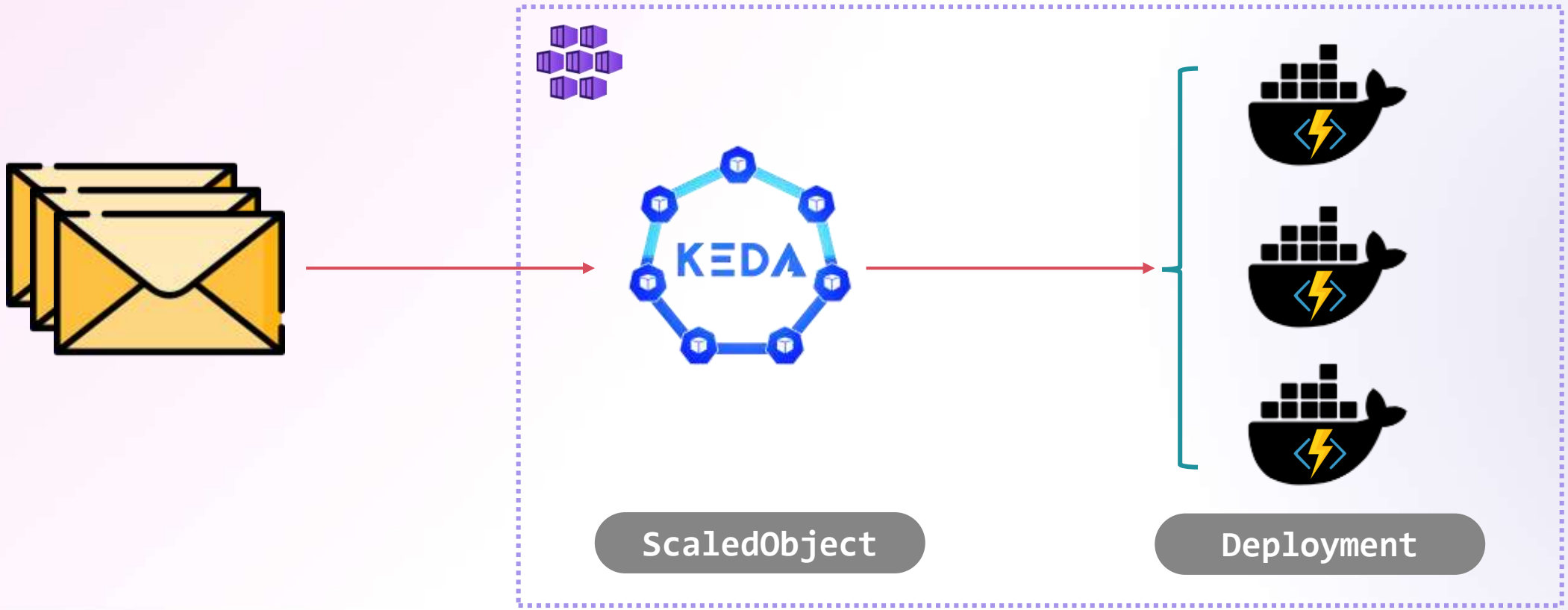


KEDA Function on AKS

Event Source

Scale Controller

Function Runtime



Hello KEDA Function

我有一個需求...

- 某：這個需求很簡單，~~怎麼實現我不管~~
- 寫一支程式
- 拿出 Storage Queue 的訊息並 print 在 Console 上

Azure Function Core Tools

- 支援本地端開發、測試及佈署工作
- 支援跨平台開發
  - Windows / macOS / Linux
- 本議程使用版本為 4.0.5455

```
$ func --version
4.0.5455
```

Azure Functions Core Tools reference

安裝方式

- Windows: MSI / winget / choco / npm
- Mac: Homebrew / npm
- Linux: apt-get / npm

Install Azure Functions Core Tools

常用的 function commands

Command context	Description
func init	Create function project
func template list	List available function templates
func new	Create function in current project
func start	Start the local function runtime
func kubernetes deploy	Deploy function project to remote Kubernetes cluster
func version	Get current version of the tool

Azure Functions Core Tools reference

建立專案

```
# Create a function project
func init {project_name} `
    --worker-runtime dotnet-isolated `
    --target-framework net8.0 `
    --docker

# Add a Dockerfile to existing project
func init --docker-only
```

Reference - func init

新增 Function

```
# Get templates list
func templates list

# Create a function
func new `
  --template {template_name} `
  --name {function_name}
```

查看可用的 Template

```
$ func templates list
C# Templates:
  Azure Blob Storage trigger
  Azure Cosmos DB trigger
  Durable Functions activity
  Durable Functions HTTP starter
  Durable Functions orchestrator
  Azure Event Grid trigger
  Azure Event Hub trigger
  HTTP trigger
  IoT Hub (Event Hub)
  Azure Queue Storage trigger
  RabbitMQ trigger
  SendGrid
  Azure Service Bus Queue trigger
  Azure Service Bus Topic trigger
  SignalR negotiate HTTP trigger
  Timer trigger
```

Template 名稱錯誤

```
$ func templates list
C# Templates:
  Azure Blob Storage trigger
```

```
$ func new `
>   --template "Azure Blob Storage trigger" `
>   --name MyAzureBlobStorageTrigger`
Template: Azure Blob Storage trigger
Function name: MyAzureBlobStorageTrigger`

Creating dotnet function...
Unknown template 'AzureBlobStoragetrigger' (Parameter 'templateName')
```

azure-functions-core-tools/issues/2955

azure-functions-core-tools/issues/3440

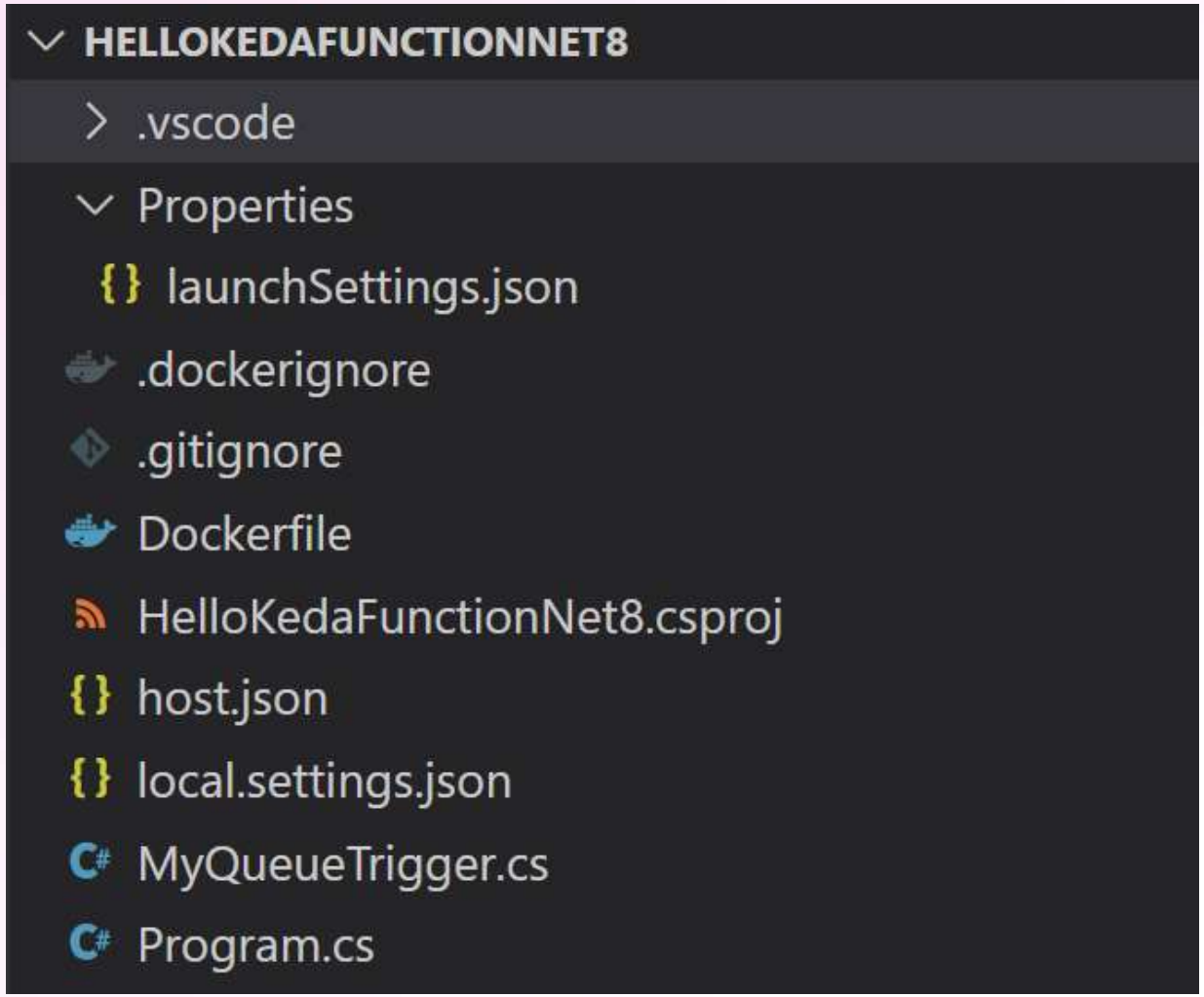
取得正確的 Template 名稱

```
$ func new --language dotnet-isolated
Use the up/down arrow keys to select a template:
QueueTrigger
HttpTrigger
BlobTrigger
TimerTrigger
EventHubTrigger
```

```
$ func new `
>   --template "Queue trigger" `
>   --name MyQueueTrigger`
Template: Queue trigger
Function name: MyQueueTrigger`

Creating dotnet function...
The function "MyQueueTrigger`" was created successfully from the
"Queue trigger" template.
```

專案目錄結構


HELLOKEDAFUNCTIONNET8
.vscode
Properties
launchSettings.json
.dockerignore
.gitignore
Dockerfile
HelloKedaFunctionNet8.csproj
host.json
local.settings.json
MyQueueTrigger.cs
Program.cs

Queue Trigger(1)

```
public class MyQueueTrigger
{
    private readonly ILogger<MyQueueTrigger> _logger;

    public MyQueueTrigger(ILogger<MyQueueTrigger> logger)
    {
        _logger = logger;
    }

    [Function(nameof(MyQueueTrigger))]
    public void Run([QueueTrigger("myqueue-items", Connection = "")]
        QueueMessage message)
    {
        _logger.LogInformation($"C# Queue trigger function
            processed: {message.MessageText}");
    }
}
```

Queue Trigger(2)

```
[Function(nameof(MyQueueTrigger))]
public void Run(
    [QueueTrigger("netconf2023", Connection = "MyStorageConn")] QueueMessage message)
{
    _logger.LogInformation($"C# Queue trigger function processed: {message.MessageText}");
}
```

```
Functions:

        MyQueueTrigger: queueTrigger

For detailed output, run func with --verbose flag.
[2023-12-02T02:47:58.567Z] Worker process started and initialized.
[2023-12-02T02:48:18.530Z] Executing 'Functions.MyQueueTrigger' (Reason='New queue message
 detected on 'netconf2023'.', Id=4f7815fa-27bd-45fc-8e1e-e2c3fab9878f)
[2023-12-02T02:48:18.539Z] Trigger Details: MessageId: a92321ba-49d1-4885-a849-e67eee0f892
c, DequeueCount: 1, InsertedOn: 2023-12-02T02:48:19.000+00:00
[2023-12-02T02:48:25.457Z] C# Queue trigger function processed: Hello .NET 8
[2023-12-02T02:48:25.491Z] Executed 'Functions.MyQueueTrigger' (Succeeded, Id=4f7815fa-27b
d-45fc-8e1e-e2c3fab9878f, Duration=7026ms)
```

Dockerfile

Dockerfile > ...

```
1   FROM mcr.microsoft.com/dotnet/sdk:8.0-preview AS installer-env
2
3   COPY . /src/dotnet-function-app
4   RUN cd /src/dotnet-function-app && \
5   mkdir -p /home/site/wwwroot && \
6   dotnet publish *.csproj --output /home/site/wwwroot
7
8   # To enable ssh & remote debugging on app service change the base image to the one below
9   # FROM mcr.microsoft.com/azure-functions/dotnet-isolated:4.0-dotnet-isolated8.0-appservice
10  FROM mcr.microsoft.com/azure-functions/dotnet-isolated:4-dotnet-isolated8.0
11  ENV AzureWebJobsScriptRoot=/home/site/wwwroot \
12      AzureFunctionsJobHost__Logging__Console__IsEnabled=true
13
14  COPY --from=installer-env ["/home/site/wwwroot", "/home/site/wwwroot"]
15
```

Deploy KEDA Function(1)

- Prerequisite
  - Azure Container Registry(ACR)
  - Azure Kubernetes Service(AKS)
    - Integrate with ACR
    - Install KEDA Operator
- 使用 `func kubernetes deploy` 指令

Deploy KEDA Function(2)

Example

```
# Deploy function container to k8s
func kubernetes deploy `
  --name {deployment_name} `
  --registry {registry_url} `
  --namespace {namespace_name}
```

Command

```
$ func kubernetes deploy `
>    --name myfirstkeda `
>    --registry netconf2023.azurecr.io `
>    --namespace my-keda
Running 'docker build -t netconf2023.azurecr.io/myfirstkeda:latest
C:\Users\rober\Desktop\test\MyFirstKedaFunction'.....done
secret/myfirstkeda created
deployment.apps/myfirstkeda created
scaledobject.keda.sh/myfirstkeda created
```

Deploy KEDA Function(3)

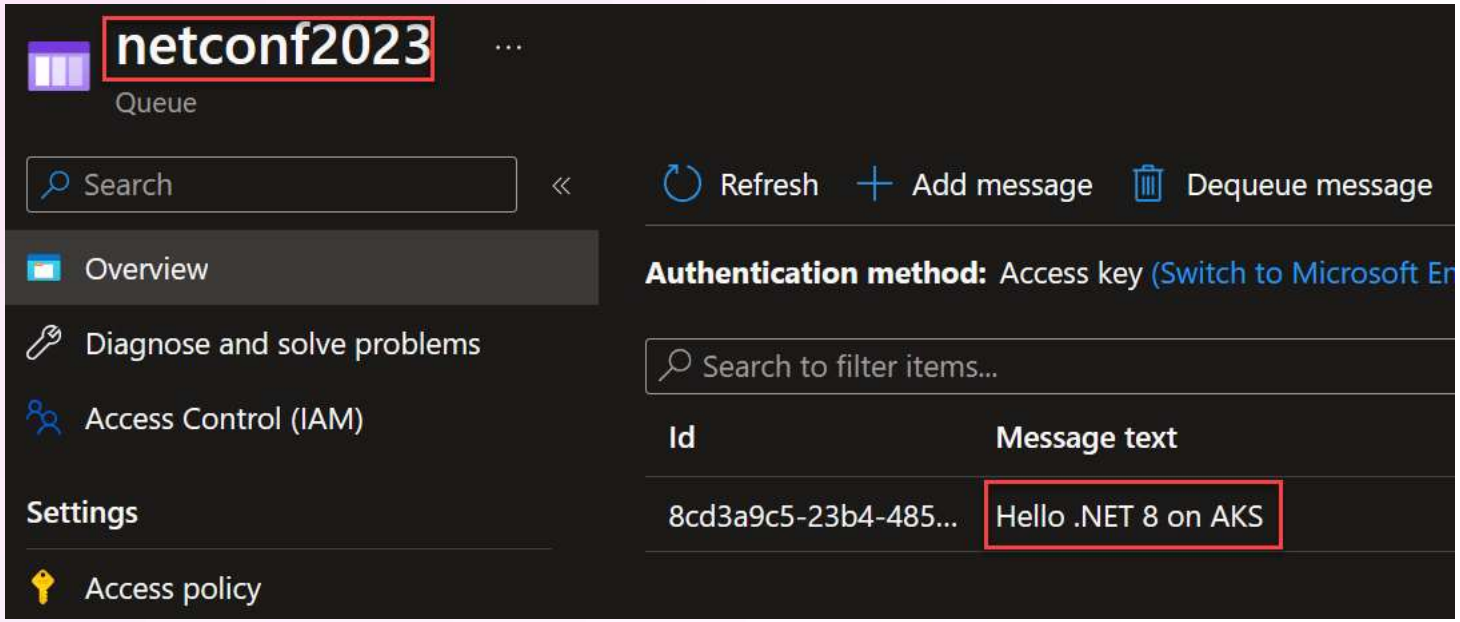
```
$ kubectl get deploy,secret,scaledobject -n my-keda
NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myfirstkeda     0/0     0            0           16m

NAME                  TYPE     DATA   AGE
secret/myfirstkeda    Opaque   3      16m

NAME                                SCALETARGETKIND      SCALETARGETNAME   MIN   MAX   TRIGGERS      AUTHENTICATION
scaledobject.keda.sh/myfirstkeda    apps/v1.Deployment   myfirstkeda                     azure-queue
```

```
$ kubectl describe scaledobject -n my-keda
Events:
  Type    Reason                    Age    From           Message
  ----    ------                    ----   ----           -------
  Normal  KEDAScalersStarted        2m55s  keda-operator  Scaler azure-queue is built.
  Normal  KEDAScalersStarted        2m55s  keda-operator  Started scalers watch
  Normal  ScaledObjectReady         2m55s  keda-operator  ScaledObject is ready for scaling
  Normal  KEDAScaleTargetDeactivated  2m55s  keda-operator  Deactivated apps/v1.Deployment my-keda/myfirstkeda from 1 to 0
```

Get message from AKS(1)




```
$ kubectl get deploy -n my-keda
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
myfirstkeda    1/1     1            1           64m
```

Get message from AKS(2)

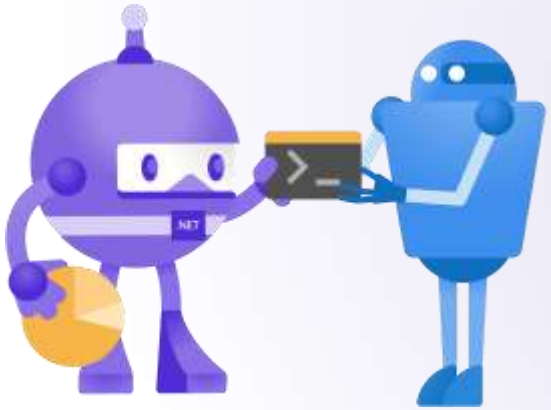
```
$ kubectl get pod -n my-keda
NAME                              READY   STATUS    RESTARTS   AGE
myfirstkeda-66db8f7c87-zmcjv      1/1     Running   0          52s
```

```
$ kubectl logs -f -n my-keda myfirstkeda-66db8f7c87-zmcjv
info: Function.MyQueueTrigger[1]
      Executing 'Functions.MyQueueTrigger' (Reason='New queue message detected on 'netconf2023'.', Id=246dcfb3-3a9c-4e
24-81e6-f047a16f98b7)
info: Function.MyQueueTrigger[0]
      Trigger Details: MessageId: 8cd3a9c5-23b4-485b-96b5-fe201ec41ead, DequeueCount: 1, InsertedOn: 2023-12-02T08:02:
08.000+00:00
info: Function.MyQueueTrigger.User[0]
      C# Queue trigger function processed: Hello .NET 8 on AKS
info: Function.MyQueueTrigger[2]
      Executed 'Functions.MyQueueTrigger' (Succeeded, Id=246dcfb3-3a9c-4e24-81e6-f047a16f98b7, Duration=129ms)
```

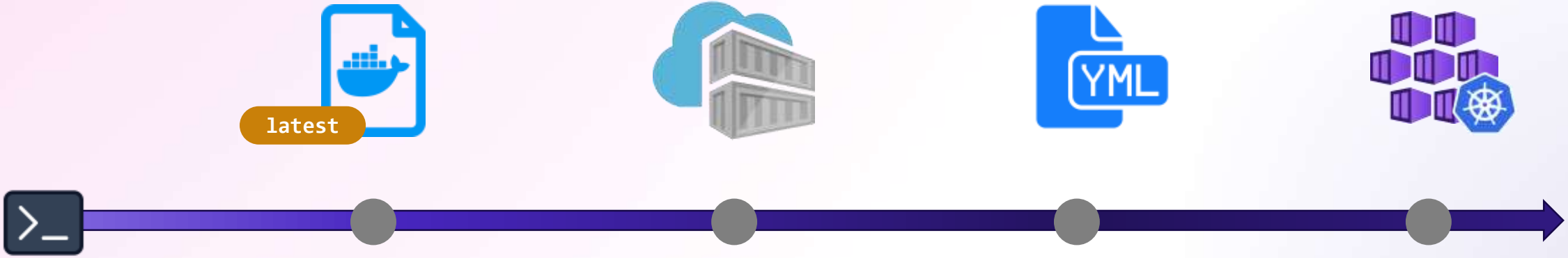
Recap – Hello KEDA Function

- 認識 Function Core Tool
- 建立一個新專案
- 新增一個 QueueTrigger Function
- 將 KEDA Function 部署到 AKS 上

Deployment Workflow



What does deploy command do?



`func kubernetes deploy`

Build

Push

Generate Manifests

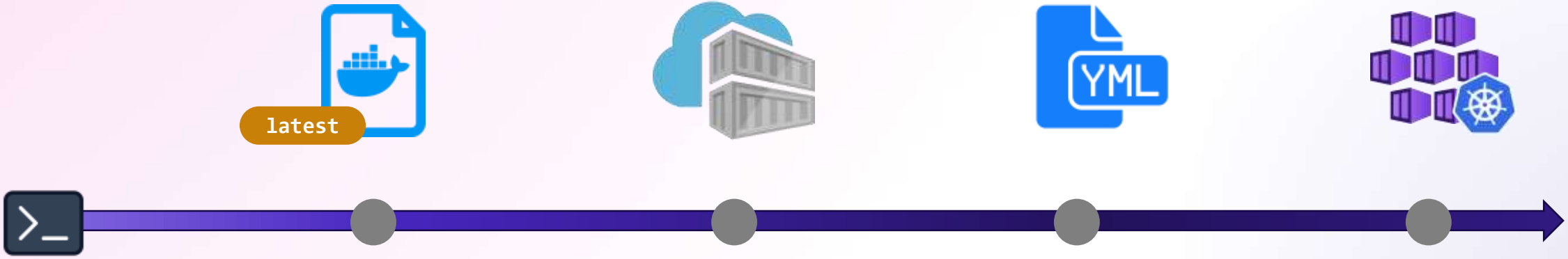
- Deployment
- Secret
- ScaledObject

Deploy

What are the problems?

- Image tag 固定為 latest
  - 不利於 artifact 的版本控制
- 預設只對 local.setting.json 做轉換
  - 多環境組態管理不易
- 整合至 CI/CD 流程需使用 function tool
  - 額外學習成本

Can we divide the workflow?



`func kubernetes deploy`

Build

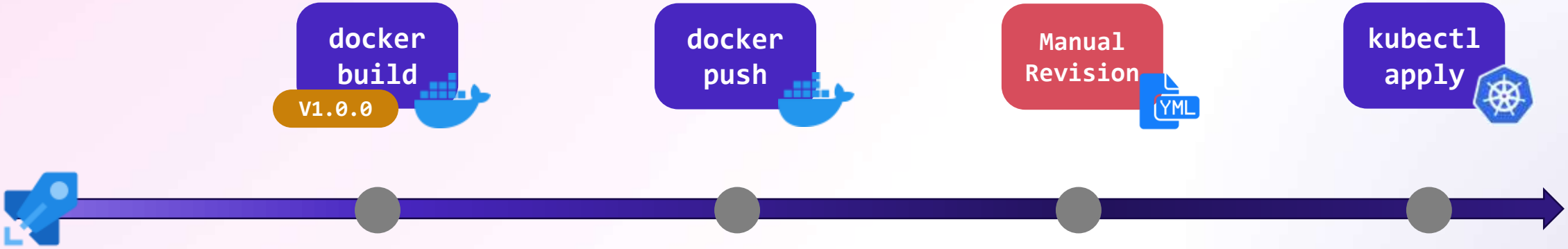
Push

Generate Manifests

- Secret
- Deployment
- ScaledObject

Deploy

Division Strategy



~~func kubernetes deploy~~

Build

Push

Generate Manifests

- Secret
- Deployment
- ScaledObject

Deploy

Output manifests as a file

- 使用 dryrun 指令將 manifest 輸出為檔案
  - 不會觸發部署流程

```
# Output manifests into a single file
func kubernetes deploy `
  --name {deployment_name} `
  --registry {registry_url} `
  --namespace {namespace_name} `
  --dry-run `
  > {output_flie}.yml
```

Secret manifest

local.setting.json

```
{
    "IsEncrypted": false,
    "Values": {
        "AzureWebJobsStorage": "UseDevelopmentStorage=true",
        "FUNCTIONS_WORKER_RUNTIME": "dotnet-isolated",
        "MyStorageConn": "DefaultEndpointsProtocol=https;Acc
    }
}
```

Base 64

secret manifest

```
apiVersion: v1
kind: Secret
metadata:
  name: myfirstkeda
  namespace: my-keda
data:
  AzureWebJobsStorage: VXNlRGV2ZWxvcG1lbnRTdG9yYWdlPXRydW
  FUNCTIONS_WORKER_RUNTIME: ZG90bmV0LWlzb2xhdGVk
  MyStorageConn: RGVmYXVsdEVuZHBvaW50c1Byb3RvY29sPWh0dHBz
```

Deployment manifest

```yaml
apiVersion: apps/v1
kind: Deployment
spec:
  # ...
  template:
    spec:
      containers:
      - name: myfirstkeda
        image: netconf2023.azurecr.io/myfirstkeda:latest
        env:
        - name: AzureFunctionsJobHost__functions__0
          value: MyQueueTrigger
        envFrom:
        - secretRef:
            name: myfirstkeda
```

Image tag

要 Enable 的 function 名稱

參考的 secret 對象

ScaledObject manifest(KEDA)

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: myfirstkeda
  namespace: my-keda
  labels: {}
spec:
  scaleTargetRef:
    name: myfirstkeda
  triggers:
  - type: azure-queue
    metadata:
      direction: In
      queueName: netconf2023
      connectionFromEnv: MyStorageConn
```

KEDA 要擴展的 Deployment 對象

Storage Queue 的資訊

當你搞清楚這一切之後...

Deployment Strategy

- 自行管理 `dryrun` 所產生的檔案內容
- 將產生的 `manifests` 檔分環境存放
- 解除對 `function command` 的依賴
  - 使用原生的 `docker` / `kubectl` 操作

調整檔案目錄結構

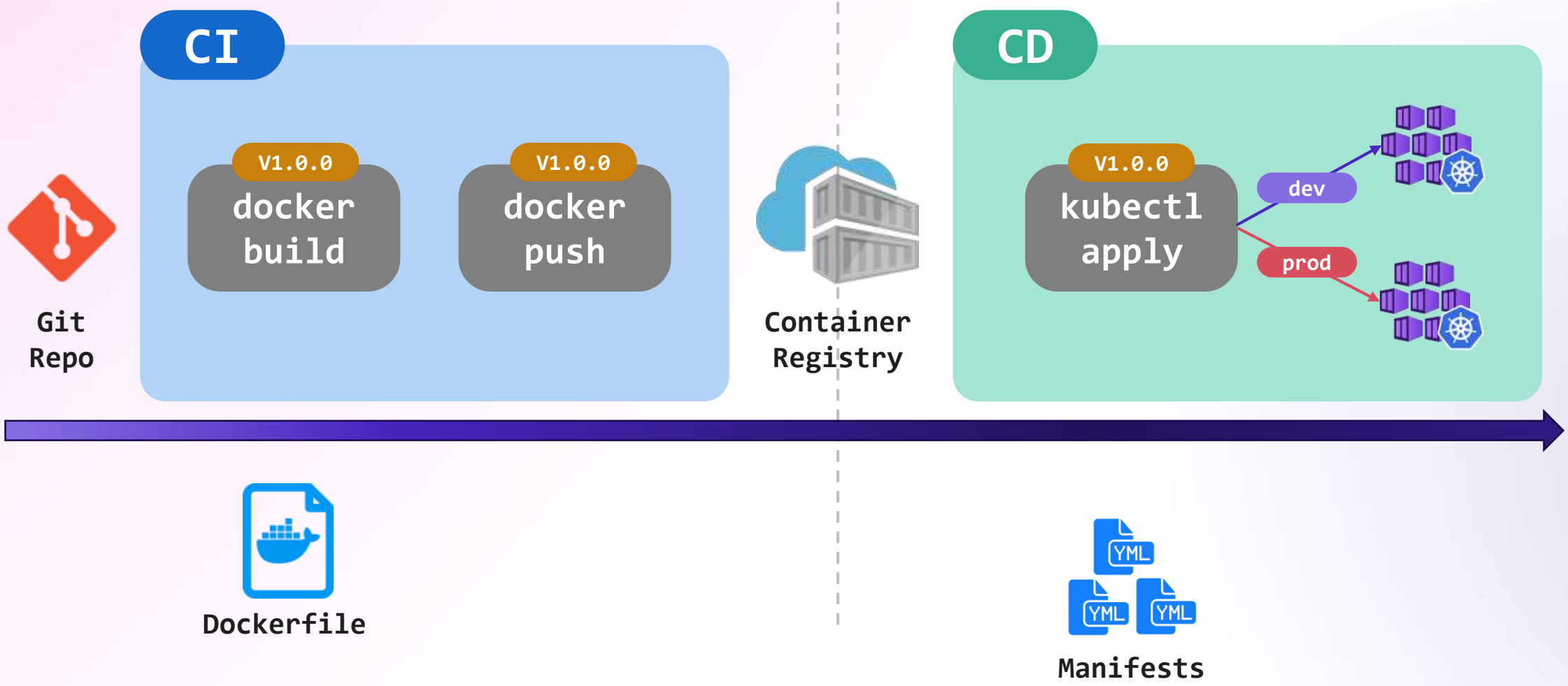
Before

```
# 根目錄
- /MyFirstKedaFunction
  - Dockerfile
  - host.json
  - local.settings.json
  - Program.cs
  - MyQueueTrigger.cs
  - MyFirstKedaFunction.csproj
```

After

```
# 根目錄
- /MyFirstKedaFunction
  # manifests 存放 YAML檔
  - /manifests
    - /dev
      - func-deployment.yml
    - /prod
      - func-deployment.yml
  # src存放 function 程式碼
  - /src
    - Dockerfile
    - host.json
    - local.settings.json
    - Program.cs
    - MyQueueTrigger.cs
    - MyFirstKedaFunction.csproj
```

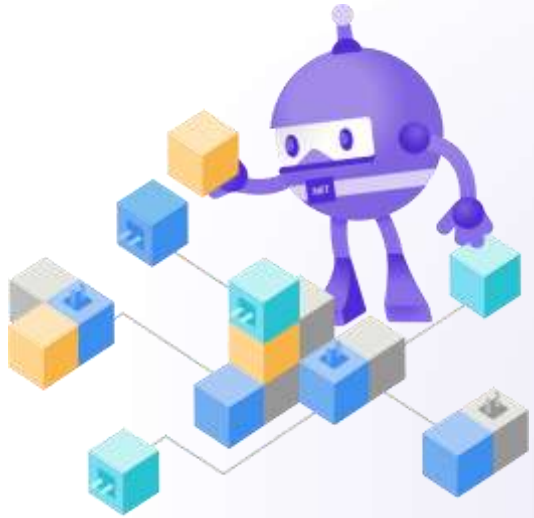
CI/CD Workflow



Recap – Deployment Workflow

- 認識 `func kubernetes deploy` 指令如何運作
  - 透過 `dryrun` 指令取得 `manifests` 檔案
- 將 `manifests` 檔依環境分放
  - `Config / Secret` 管理可依實務流程調整
- 重新調整部署流程

Other Tips



Configure Max Replicas

- 預設 maxReplicaCount 為 100
- 建議依環境及可用資源設定上限值
  - 避免資源耗盡
    - CPU / Memory / Available Number of pods

KEDA Scaling Deployments

加入 appsettings(1)

- 讓原生 function 也可以支援多環境 config 的需求
  - 適合存放 non-sensitive 的設定值
- 需額外安裝 Nuget 套件
  - Microsoft.Azure.Functions.Extensions
- 環境變數使用 AZURE_FUNCTIONS_ENVIRONMENT
  - 請不要用成 ASPNETCORE_ENVIRONMENT

Customizing configuration sources

加入 appsettings(2)

Program.cs

```
var host = new HostBuilder()
    .ConfigureAppConfiguration((context, config)=>
    {
        var env = context.HostingEnvironment.EnvironmentName;
        config
            .AddJsonFile(Path.Combine("appsettings.json"), optional: false, reloadOnChange: false)
            .AddJsonFile(Path.Combine($"appsettings.{env}.json"), optional: true, reloadOnChange: false)
            .AddEnvironmentVariables();
    })
    .ConfigureFunctionsWorkerDefaults()
    .Build();
```

MyQueueTrigger.cs

```
public class MyQueueTrigger
{
    public MyQueueTrigger(IConfiguration _configuration) { }

    //...
}
```

Customizing configuration sources

Binding Expression

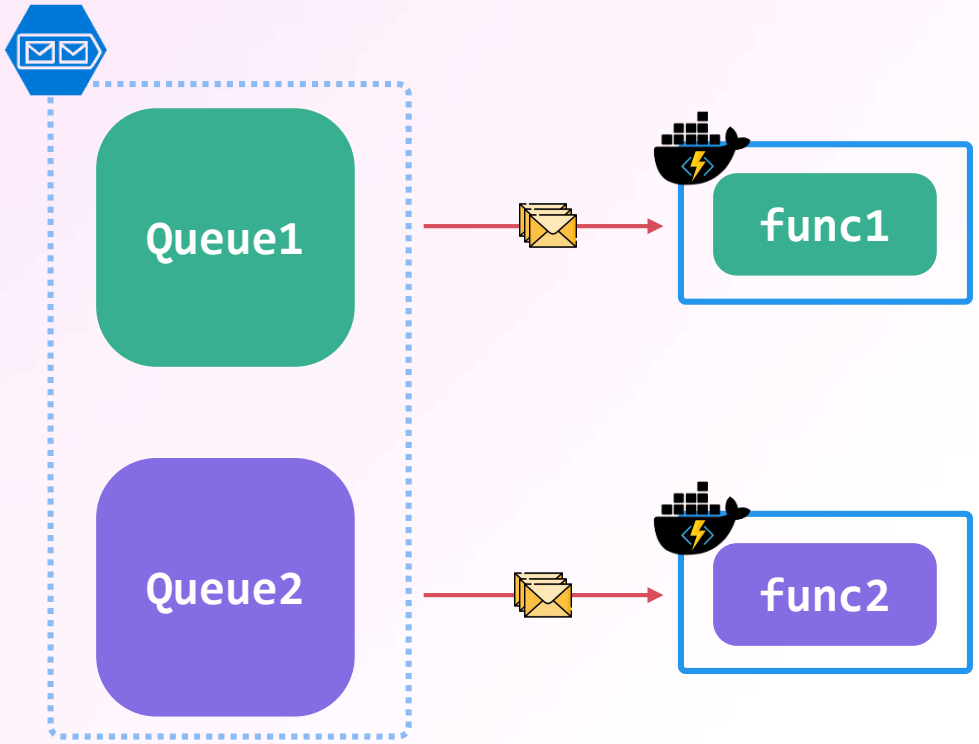
- Bind expression 無法從 appsettings 取值
  - 只能從環境變數或 local.setting.json
- 一般參數在前後要加上 % 符號
  - Connection 是例外

```
[Function(nameof(MyQueueTrigger))]
public void Run(
    [QueueTrigger("%QueueName%", Connection = "MyStorageConn")] QueueMessage message)
{
    _logger.LogInformation($"[netconf2023]: C# Queue trigger function processed: {message.MessageText}");
}
```

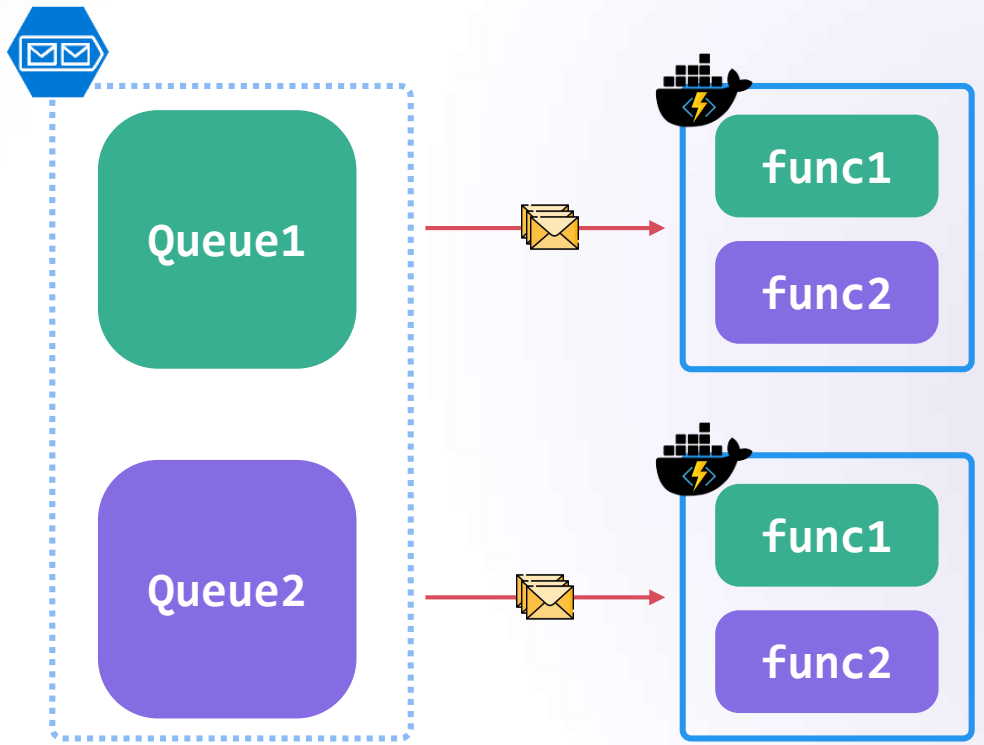
Functions bindings expressions patterns

Multi Event-based Functions

Functions from different container

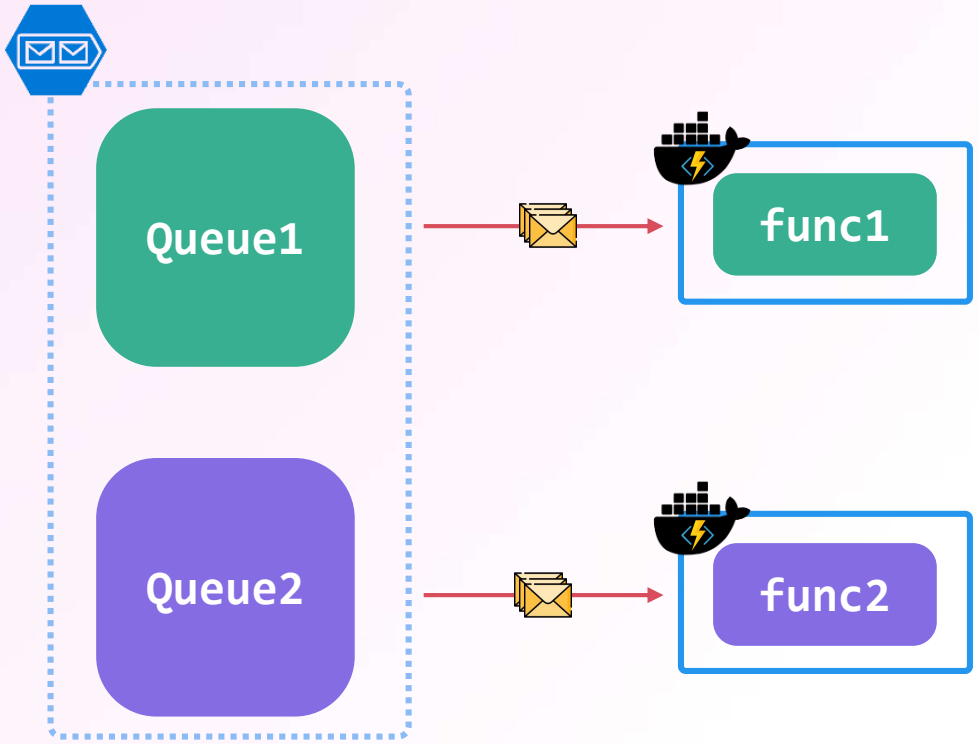


Functions from the same container

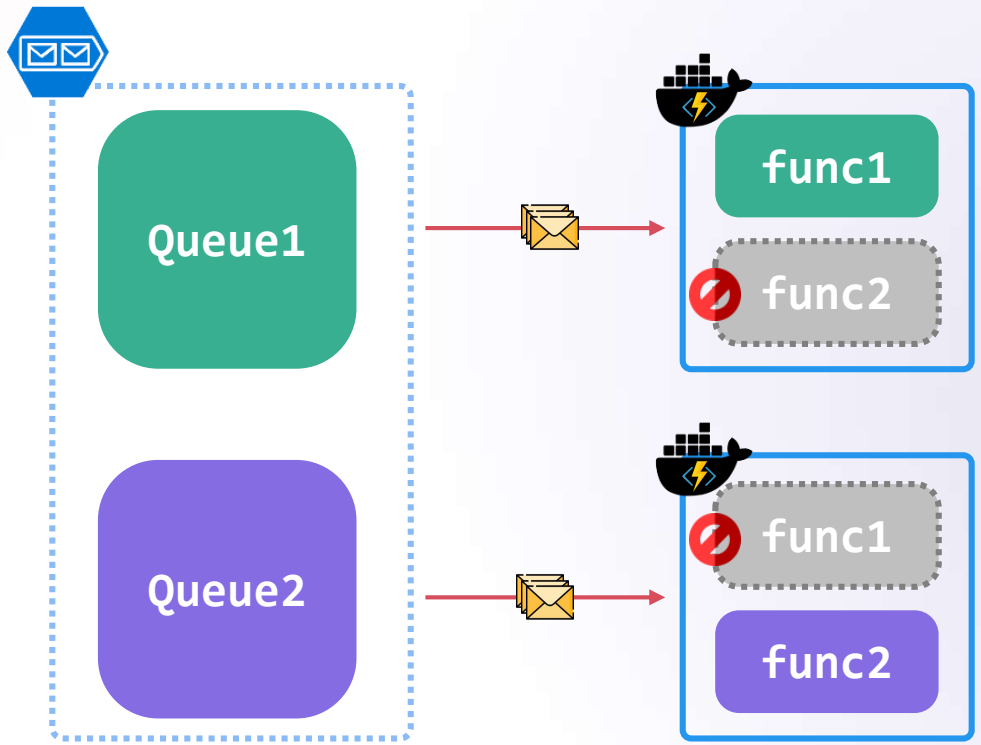


Multi Event-based Functions

Functions from different container



Functions from the same container



Control Function Toggle

- 內建有兩種 feature toggle 方式
- local.settings.json
  -
  -

```
{
    "Values": {
        "AzureWebJobs.{function_1_name}.Disabled": false,
        "AzureWebJobs.{function_2_name}.Disabled": true
    }
}
```

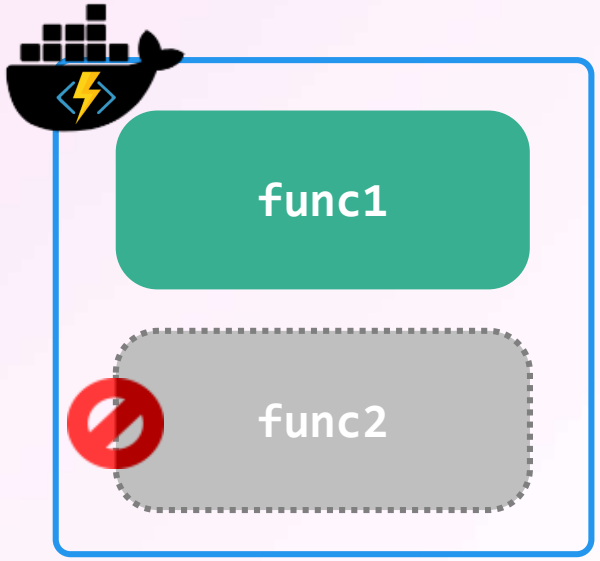
- host.json

```
{
    "functions": [ "{function_1_name}", "{function_2_name}" ]
}
```

host.json - run functions

Disable Function from app settings

Example - Disable func2



host.json

```
env:
# Enable by host.json
- name: AzureFunctionsJobHost__functions__0
  value: func1
```

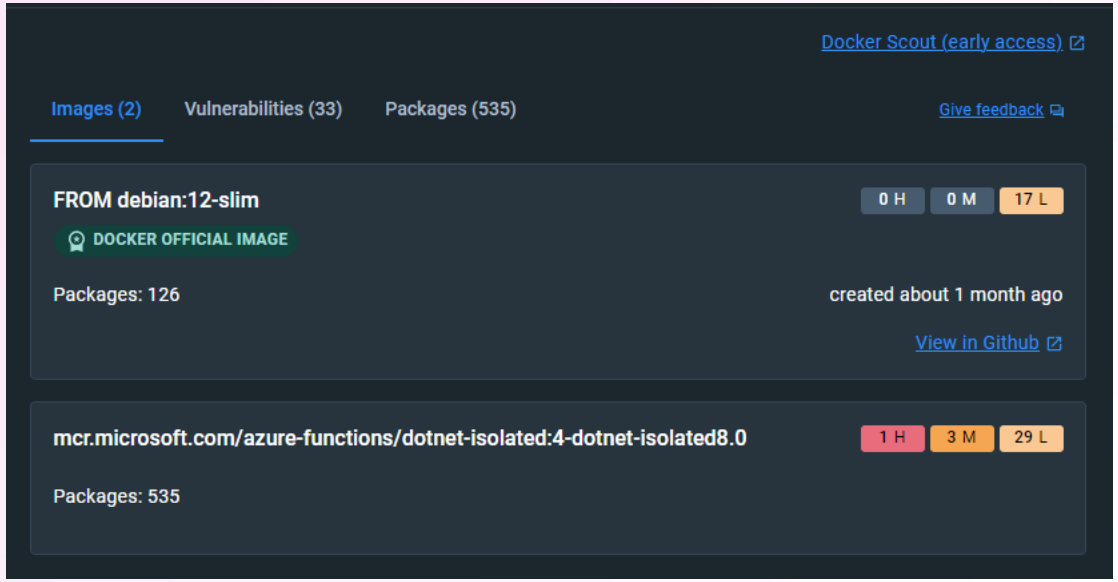
local.settings.json

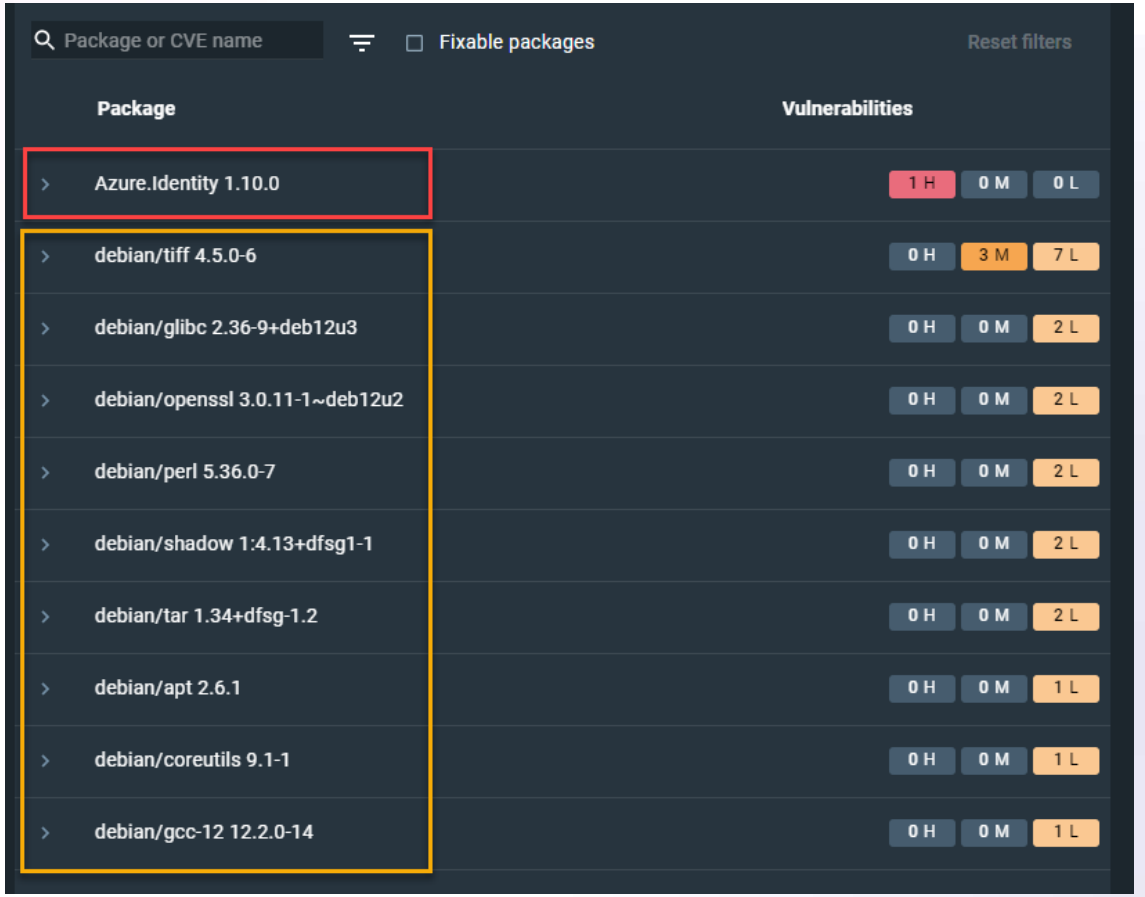
```
env:
# Disable by local.settings.json
- name: AzureWebJobs.func2.Disabled
  value: true
```

Docker Image Security

- Image 本身被掃出太多 CVE 漏洞
- 採用 Debian 11/12 做為 Base Image
- 洞從哪來?
  - OS 本身內建安裝套件
  - Function SDK 相依套件

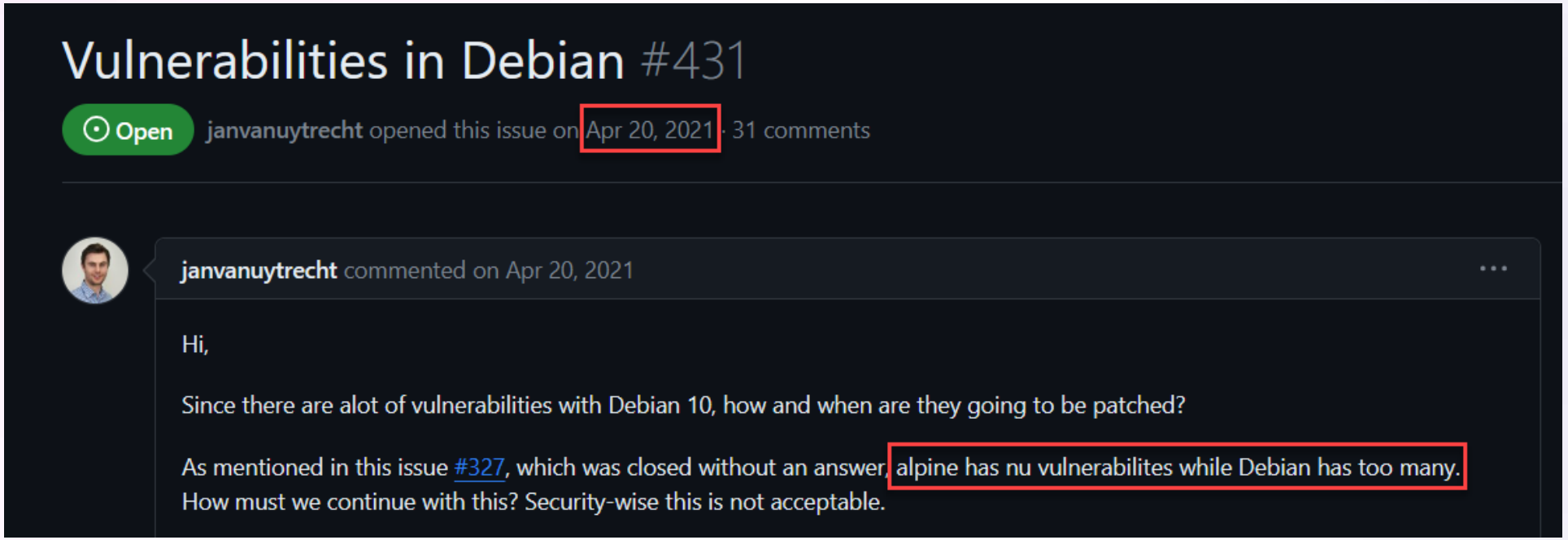
Scanning by Docker Scout





如果...我們很在意資安?

- 有網友建議官方提供 `alpine` 版本
  - 尚未得到官方回應 **404 Not Found**



有什麼解決方案?

- DIY Alpine: Building functions base images on demand
- 採用 nightly build 版本的 image

Function 的 Storage 有何用途?

- 主要用來保存特定 function 的狀態
- 當設定 `"AzureWebJobsStorage": "UseDevelopmentStorage=true"`
  - 會嘗試連線到本機的 Storage Emulator(Azurite)
- 多個 function 可共享同個 storage
- 官方不建議在正式環境共享 storage
  - 目的是避免 Function Host ID 碰撞問題

Azure Function Storage Considerations

什麼是 Host ID?

- 一個 Function 的唯一識別碼
- 長度至多僅能 32 個字元
- 在 4.x 版後碰撞會導致 function crash

Azure Function Host ID Collision

Azurite Storage Emulator

- 可於本地端模擬 Blob / Queue / Table 行為
  - 須保留 10000 / 10001 / 10002 ports 供使用
- 支援多種安裝方式: VS / VS Code / npm / Docker

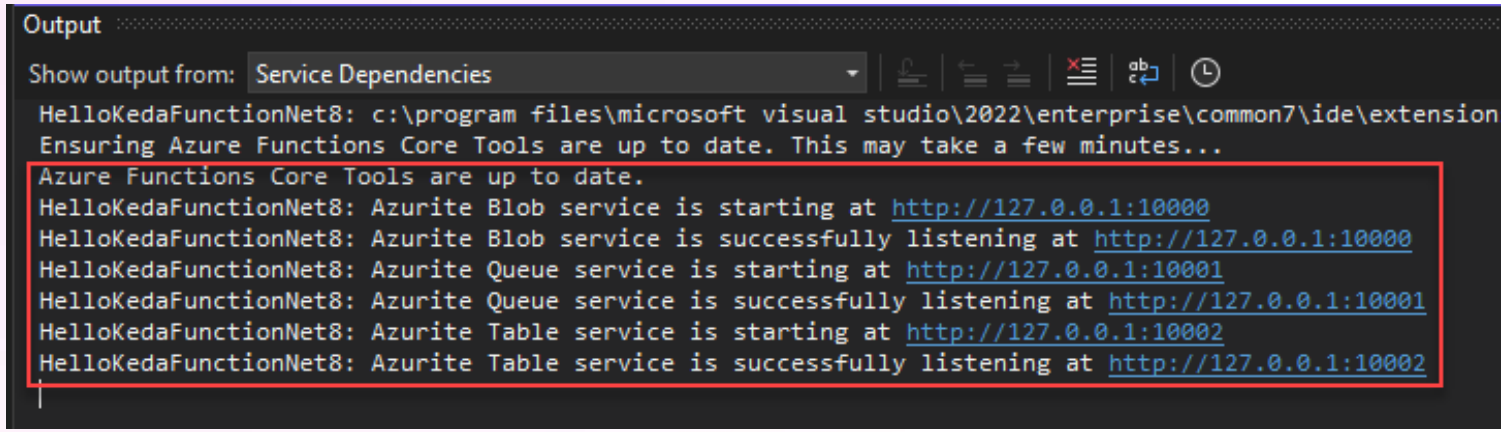
Use the Azurite emulator for local Azure Storage development

KEDA Function 的 Storage 設計

- Host ID 會從 Pod 名稱截取前32字元
  - Deployment 名稱不要太長
- 將 Azurite 跟 Function 放在同一個 Pod 裡面
  - 無須額外管理 Azurite 的狀態
- 若要採用 Shared Storage Account 的機制
  - 要設定 lifecycle policy 來清理無用的 function host 資訊

Visual Studio with Azurite

- 使用 VS 打開 function 專案就會啟動
- VS 關閉後，Azurite process 並不會被回收



Conclusion

- DX > OX
  - 開發便利、維運複雜
- 運行環境的資源可控性高
  - 效能、內網
- KEDA 使 function 的用途更加放大

Thank you

https://aka.ms/get-dotnet-8