使用 OpenTelemetry 的 .NET 可觀察性 (安裝篇)

前言

本篇文章主題為

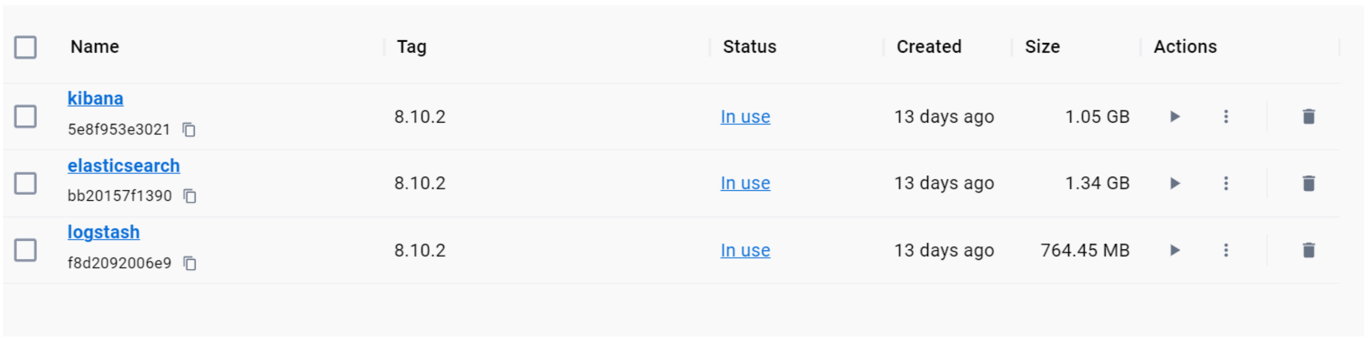
1. 使用docker安裝ELK
2. 使用Auto instrumentation 於 .net
3. 使用Auto instrumentation 於 現有IIS的專案

本文撰寫時的OTel版本為: `1.25`

ELK 安裝

本次使用Docker安裝ELK。本次使用的版本如下

- `Kibana: 8.10.2`
- `ElasticSearch: 8.10.2`
- `Elastic-Agent: 8.10.2`



Name	Tag	Status	Created	Size	Actions
kibana 5e8f953e3021	8.10.2	In use	13 days ago	1.05 GB	
elasticsearch bb20157f1390	8.10.2	In use	13 days ago	1.34 GB	
logstash f8d2092006e9	8.10.2	In use	13 days ago	764.45 MB	

透過下方docker-compose內容，並開啟`CMD`後輸入以下指令

```
docker-compose pull #下載image
docker-compose up #啟動container
```

docker-compose.yml內容如下

```
version: "3.8"
services:
  elasticsearch:
     image: elasticsearch:8.10.2
     environment:
      - discovery.type=single-node
      - network.host=0.0.0.0
      - http.host=0.0.0.0
      - xpack.security.enabled=true
      - xpack.security.authc.api_key.enabled=true
```

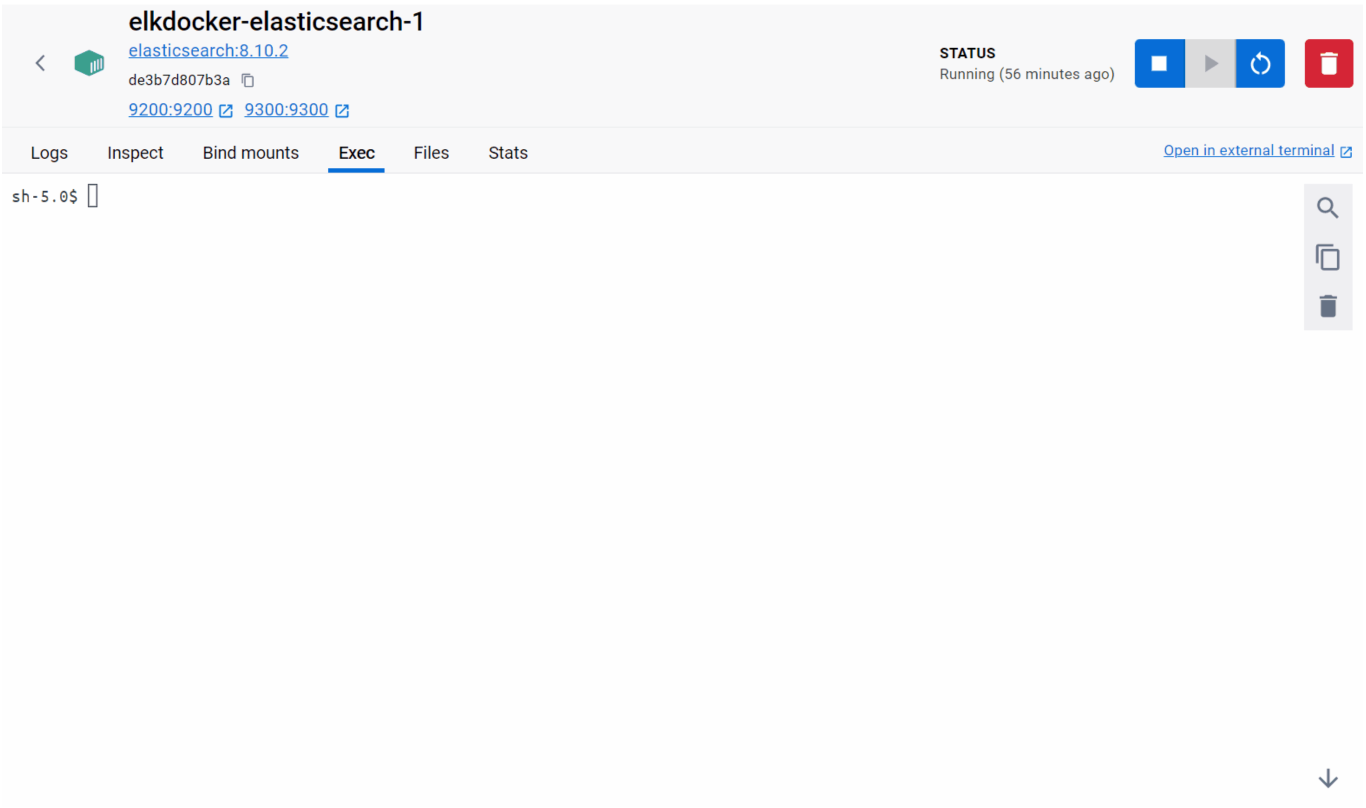
```
         - ELASTIC_PASSWORD=changeme
        ports:
          - 9200:9200
          - 9300:9300
        healthcheck:
          test: nc -z localhost 9200 || exit 1
          interval: 5s
          timeout: 10s
          retries: 100
     kibana:
        image: kibana:8.10.2
        ports:
          - 5601:5601
        environment:
          - ELASTICSEARCH_USERNAME="kibana_system"
          - ELASTICSEARCH_PASSWORD="kibana_system"
        healthcheck:
          test: ["CMD-SHELL", "curl -u kibana_system:kibana_system -s
http://localhost:5601/api/status"]
          interval: 5s
          timeout: 10s
          retries: 120
        depends_on:
         elasticsearch:
           condition: service_healthy
     fleet-server:
       image: elastic/elastic-agent:8.10.2
       container_name: fleet-server
       user: root
       ports:
          - 8220:8220
       environment:
         - FLEET_SERVER_ENABLE=1
         - FLEET_SERVER_ELASTICSEARCH_HOST=http://elasticsearch:9200
         - FLEET_SERVER_SERVICE_TOKEN=<TOKEN>
         - FLEET_SERVER_POLICY_ID=fleet-server-policy
         - FLEET_SERVER_ELASTICSEARCH_USERNAME=elastic
         - FLEET_SERVER_ELASTICSEARCH_PASSWORD=elastic
         - p 8220:8220
       healthcheck:
         test: ["CMD-SHELL", "curl -u elastic:elastic -s
http://localhost:5601/api/status"]
       depends_on:
         kibana:
           condition: service_healthy
     agent01:
       image: elastic/elastic-agent:8.10.2
       container_name: agent01
       user: root
       environment:
         - FLEET_ENROLLMENT_TOKEN=<TOKEN>
         - FLEET_ENROLL=1
         - FLEET_URL=https://fleet-server:8220
         - FLEET_INSECURE=true
         - p 8200:8200
       ports:
         - 8200:8200
       depends_on:
         fleet-server:
           condition: service_healthy
```

下載後container並未全部啟動，須完成下方流程。

1. 進入ElasticSearch設定kibana的帳號密碼，首先進入elasticsearch container輸入以下指令

```
./bin/elasticsearch-setup-passwords interactive
```

可參考下方的gif圖，設定的密碼之後會用到，因此需要先記下來。

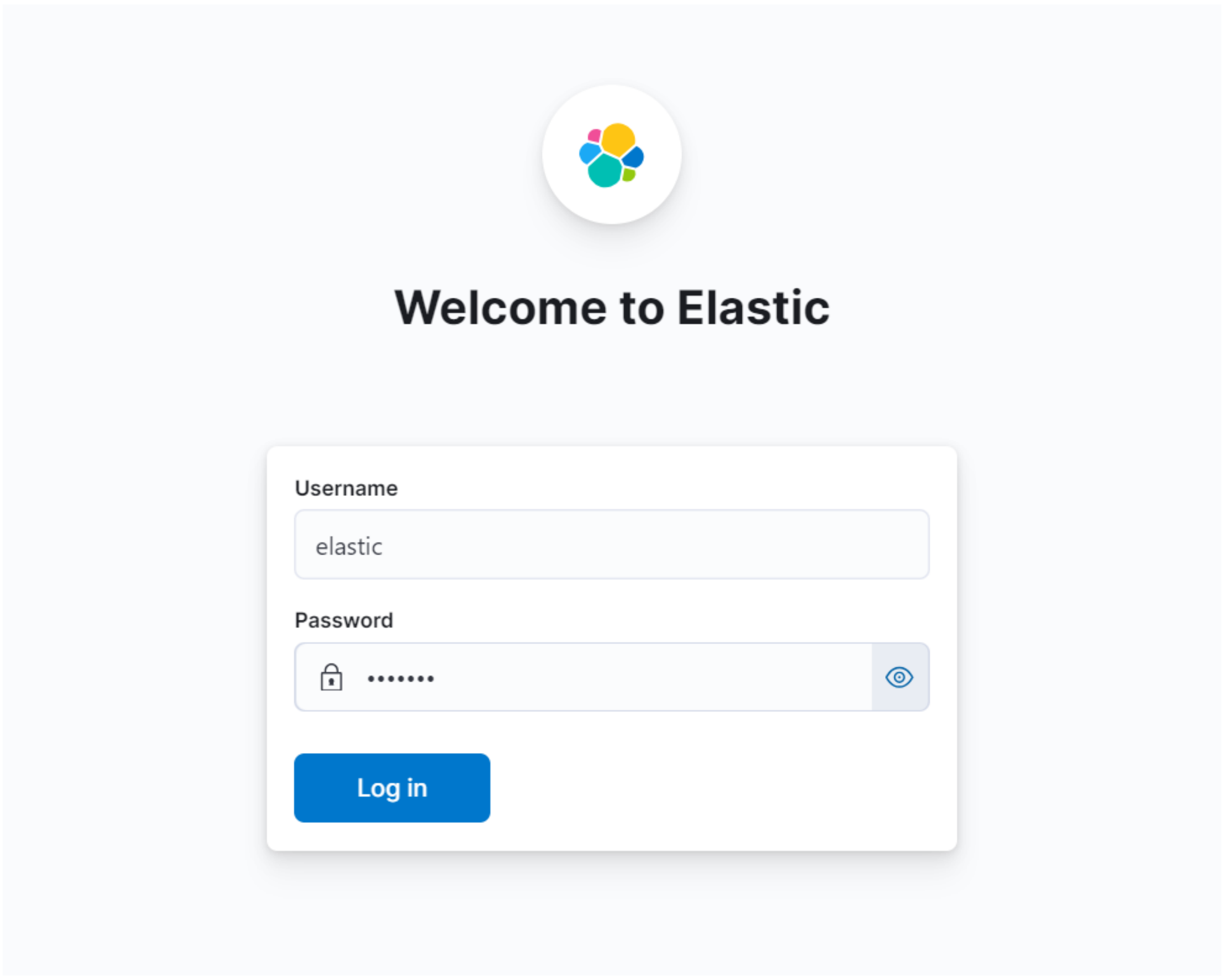


```
sh-5.0$
```

設定完`kibana_system`的密碼後，需要把密碼設定回`docker-compose.yml/kibana.ELASTICSEARCH_PASSWORD`的欄位

之後重跑一次`docker-compose up`

啟動後可以看到docker container如下，並且可以透過`localhost:5601`進入Kibana



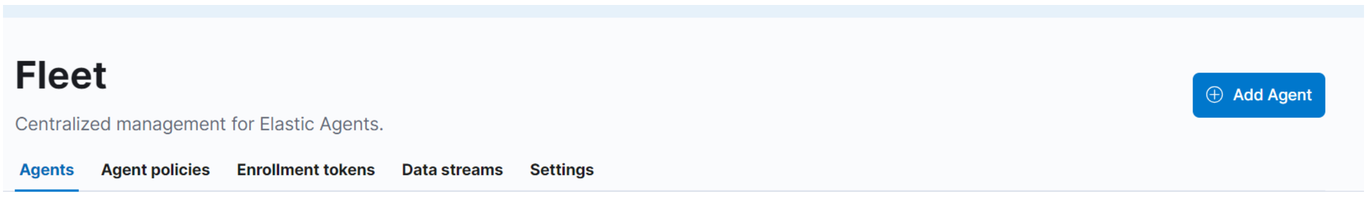
一進到頁面後需要輸入`elastic`的帳號才能進入，密碼為剛剛設定的密碼。

若遇到無法登入時需要進入到ES的container內修改密碼，指令如下

```
./elasticsearch-reset-password -u elastic
```

此指令代表要重設elastic這user的密碼，重設完後會得到新的密碼，輸入即可登入。

登入之後需要先設定fleet-server的policy，點選左方的menu選擇Management選區的`Fleet`



Centralized management for Elastic Agents.

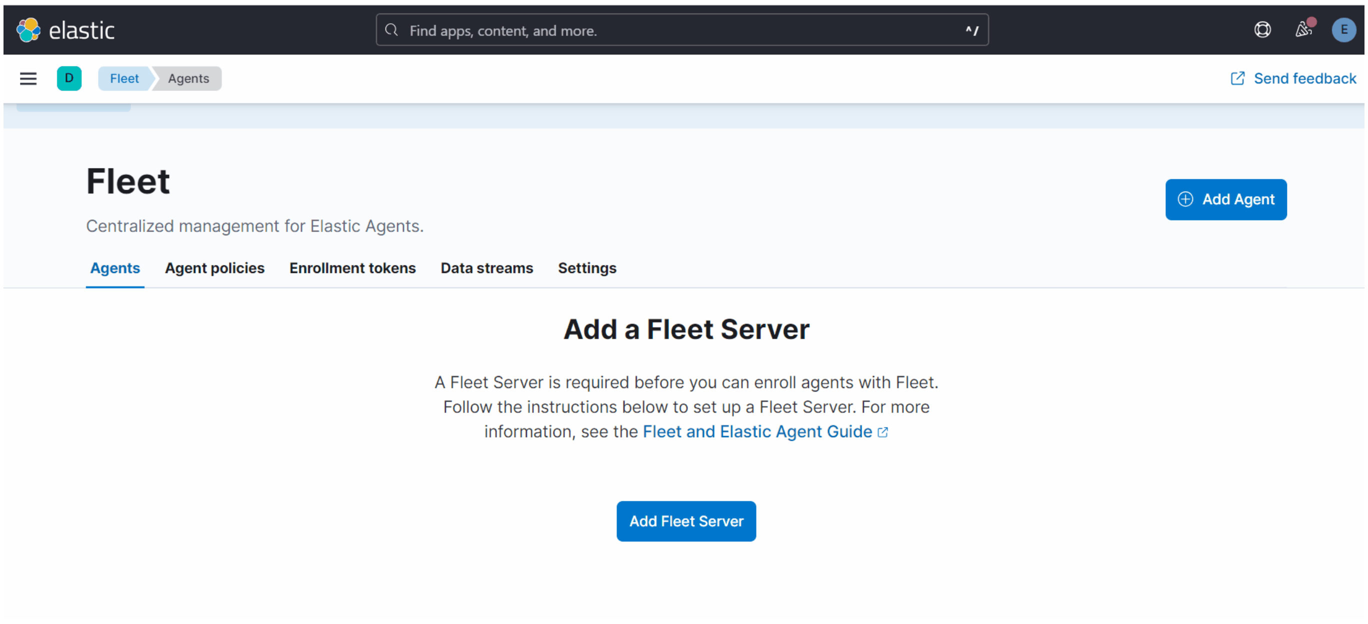
Add a Fleet Server

A Fleet Server is required before you can enroll agents with Fleet. Follow the instructions below to set up a Fleet Server. For more information, see the Fleet and Elastic Agent Guide

接著請照以下流程設定

1. 選擇 `Add Fleet Server`
2. 選擇`Advance`
3. 然後點選`Create Polity`
4. 將token貼回`docker-compose.yml/fleet-server.FLEET_SERVER_SERVICE_TOKEN`的欄位
5. 當出現`Agent Policy Created`出現時，代表有成功。
6. 產生`service-token`
7.
8. 重啟`docker-compose up`
9. 回到網頁確認是否有連上fleet-server

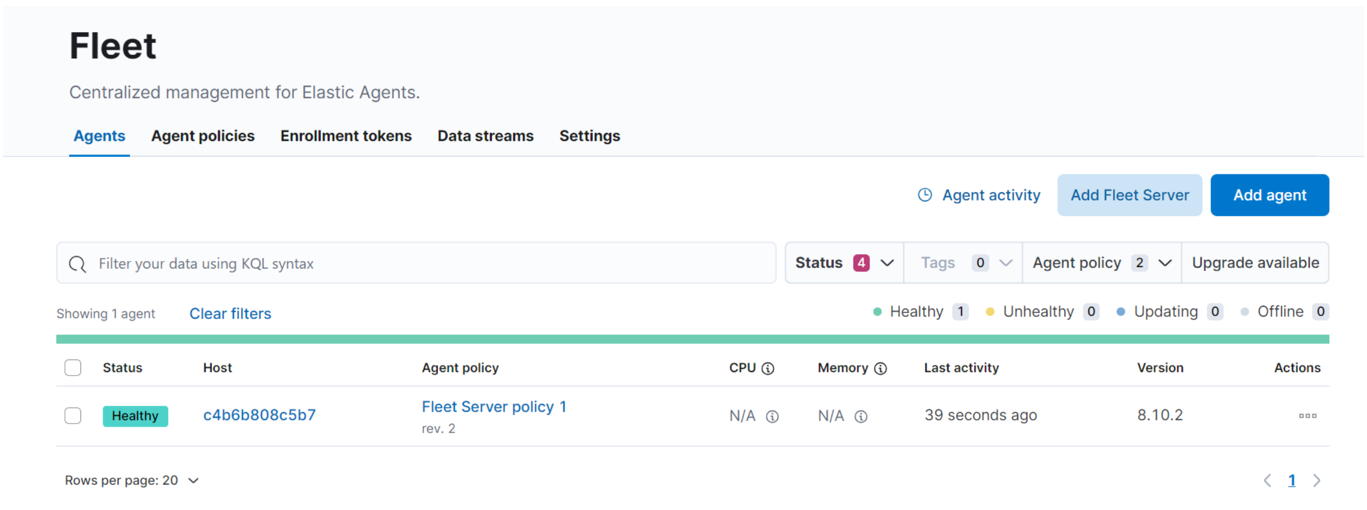
流程可參考以下示範



Centralized management for Elastic Agents.

A Fleet Server is required before you can enroll agents with Fleet. Follow the instructions below to set up a Fleet Server. For more information, see the Fleet and Elastic Agent Guide

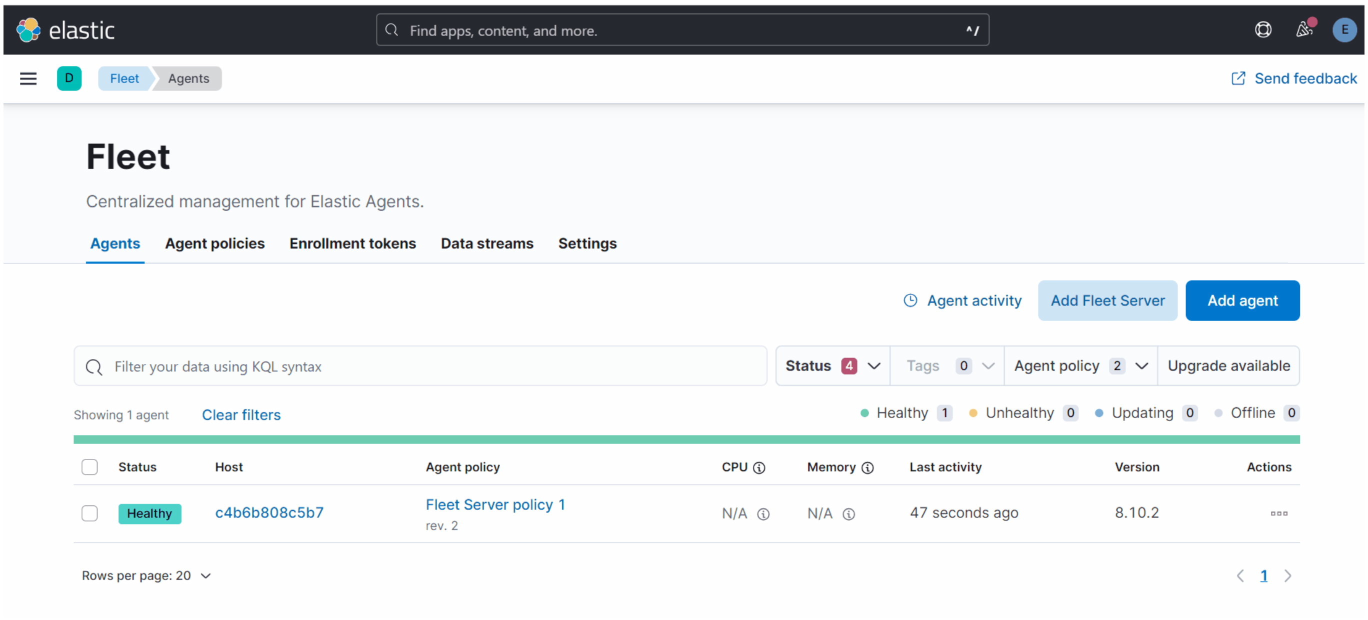
接著需要設定agent的policy,



Status	Host	Agent policy	CPU	Memory	Last activity	Version	Actions
Healthy	c4b6b808c5b7	Fleet Server policy 1 rev. 2	N/A	N/A	39 seconds ago	8.10.2	

1. 點選`Add Agent`
2. `Create Policy`
3. 在下方的Enroll找到`FLEET_ENROLLMENT_TOKEN`並貼回docker-compose裡面`agent01.FLEET_ENROLLMENT_TOKEN`的欄位
4. 重啟`docker-compose up`
5. 回到網頁確認是否有連上agent

可參考以下流程

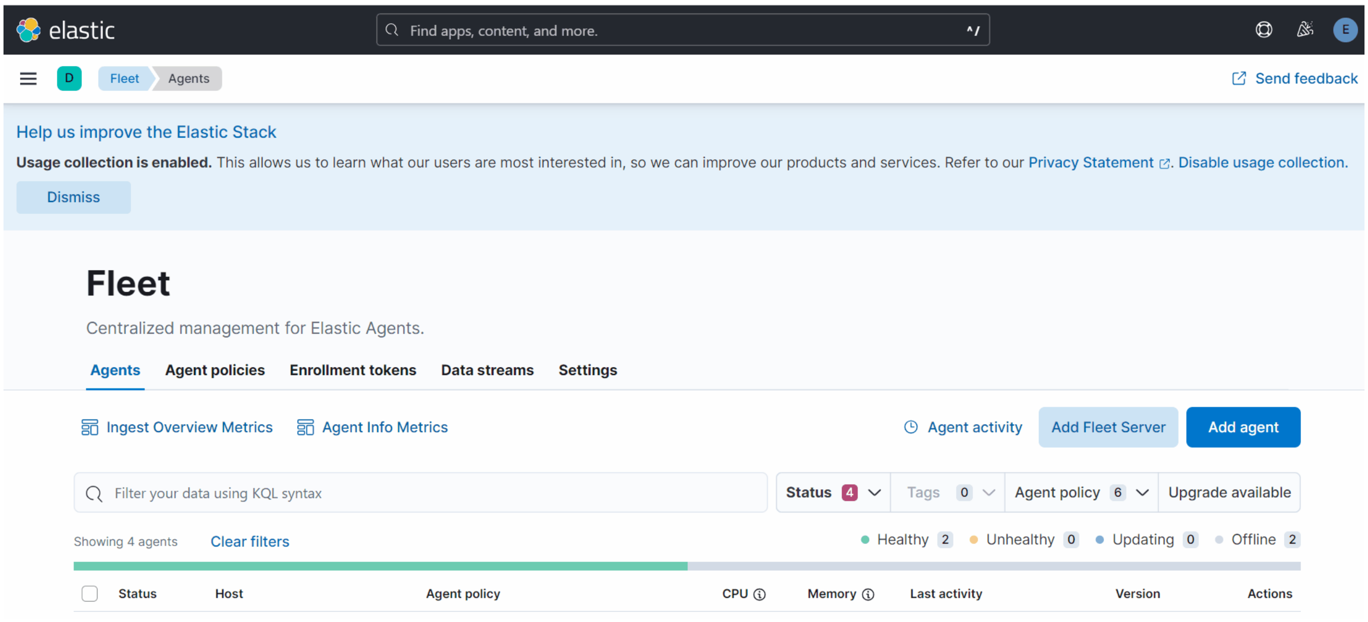


Status	Host	Agent policy	CPU	Memory	Last activity	Version	Actions
Healthy	c4b6b808c5b7	Fleet Server policy 1 rev. 2	N/A	N/A	47 seconds ago	8.10.2	

接著要設定APM server，點選左方的menu選擇Management選區的`APM`

1. 選擇 `Add Data`
2. 選擇`Manage APM integration in Fleet`
3. `Add Elastic APM`
4. 需要將Host欄位的`localhost`改為 `0.0.0.0`
5. `Save and Continue`
6. 成功後選擇 `Add Elastic Agent`
7. 一樣複製`ENROLLMENT_TOKEN`並貼回docker-compose裡面`agent01.FLEET_ENROLLMENT_TOKEN`的欄位
8. 重啟`docker-compose`
9. 回到網頁確認是否有連上agent
10. 接著點進Agent裡面頁面旁邊的Setting，需要把elastissearch的網址從`localhost`改為`elasticsearch`

到此即設定完成，接著進management申請完API Key後，完成 `安裝OpenTelemetry 於.net`的步驟，即能看到APM資料，可參考以下流程。



elastic

Find apps, content, and more.

D Fleet Agents

Send feedback

Help us improve the Elastic Stack

Usage collection is enabled. This allows us to learn what our users are most interested in, so we can improve our products and services. Refer to our Privacy Statement. Disable usage collection.

Dismiss

Fleet

Centralized management for Elastic Agents.

Agents | Agent policies | Enrollment tokens | Data streams | Settings

Ingest Overview Metrics | Agent Info Metrics | Agent activity | Add Fleet Server | Add agent

Filter your data using KQL syntax | Status 4 | Tags 0 | Agent policy 6 | Upgrade available

Showing 4 agents | Clear filters | ● Healthy 2 | ● Unhealthy 0 | ● Updating 0 | ● Offline 2

	Status	Host	Agent policy	CPU	Memory	Last activity	Version	Actions

使用Auto instrumentation 於 .net

以下安裝流程為參考Otel官方的教學文件

1. 先建置測試專案

```
dotnet new web
```

2. 專案`Program.cs`內容

```
using System.Globalization;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

var logger = app.Logger;

int RollDice()
{
    return Random.Shared.Next(1, 7);
}

string HandleRollDice(string? player)
{
    var result = RollDice();

    if (string.IsNullOrEmpty(player))
    {
        logger.LogInformation("Anonymous player is rolling the dice: {result}",
result);
```

```
    }
    else
    {
        logger.LogInformation("{player} is rolling the dice: {result}", player,
result);
    }

    return result.ToString(CultureInfo.InvariantCulture);
}

app.MapGet("/rolldice/{player?}", HandleRollDice);

app.Run();
```

3. 修改 properties/launchSetting.json

```
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "http://localhost:8080",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

4. 使用auto-instrumentation的方式置入instrumentation

開啟power-shell (需要管理員權限)，並執行以下指令

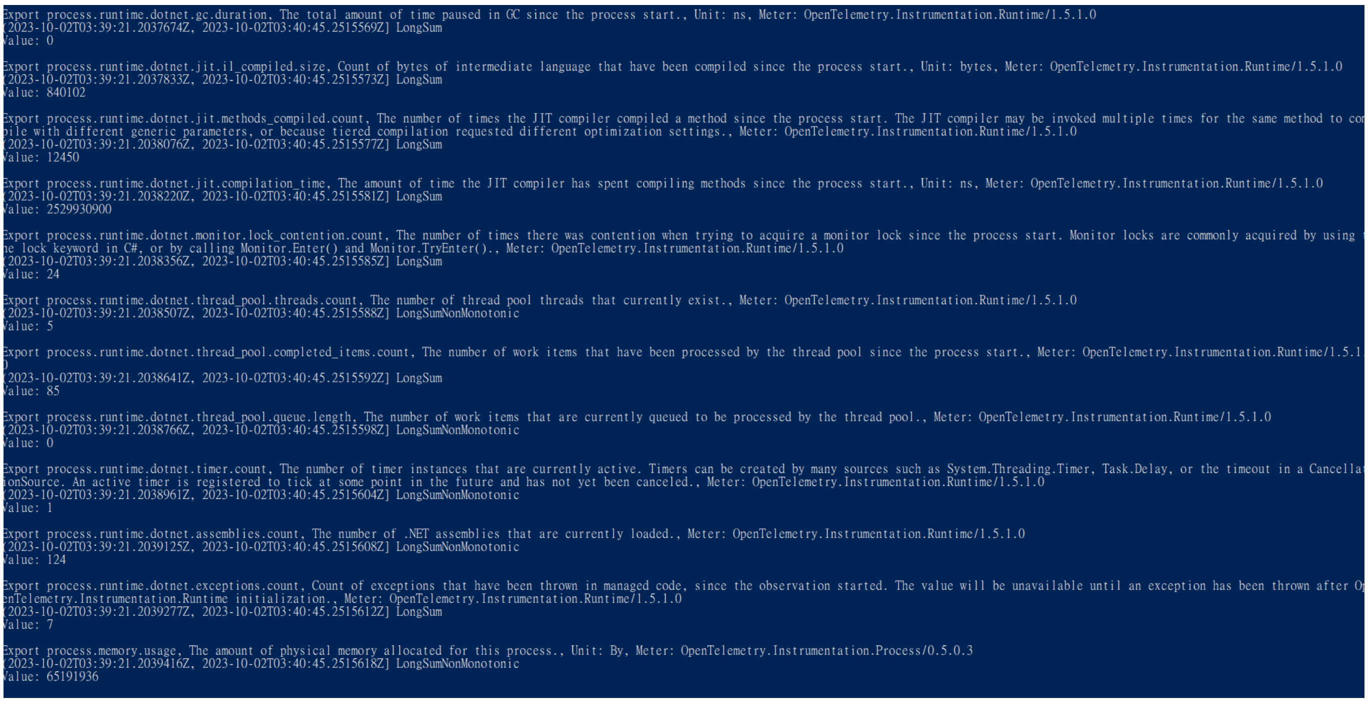
```
$module_url = "https://github.com/open-telemetry/opentelemetry-dotnet-
instrumentation/releases/latest/download/OpenTelemetry.DotNet.Auto.psm1"
$download_path = Join-Path $env:temp "OpenTelemetry.DotNet.Auto.psm1"
Invoke-WebRequest -Uri $module_url -OutFile $download_path -UseBasicParsing
Import-Module $download_path
Install-OpenTelemetryCore
$env:OTEL_TRACES_EXPORTER="none"
$env:OTEL_METRICS_EXPORTER="none"
$env:OTEL_LOGS_EXPORTER="none"
$env:OTEL_DOTNET_AUTO_TRACES_CONSOLE_EXPORTER_ENABLED="true"
$env:OTEL_DOTNET_AUTO_METRICS_CONSOLE_EXPORTER_ENABLED="true"
```

```
$env:OTEL_DOTNET_AUTO_LOGS_CONSOLE_EXPORTER_ENABLED="true"
Register-OpenTelemetryForCurrentSession -OTelServiceName "RollDiceService"
```

最後執行專案

```
dotnet run
```

執行完後可以看到console內有openTelemetry的log



```
Export process.runtime.dotnet.gc.duration, The total amount of time paused in GC since the process start., Unit: ns, Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2037674Z, 2023-10-02T03:40:45.2515569Z] LongSum
Value: 0

Export process.runtime.dotnet.jit.il_compiled.size, Count of bytes of intermediate language that have been compiled since the process start., Unit: bytes, Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2037833Z, 2023-10-02T03:40:45.2515573Z] LongSum
Value: 840102

Export process.runtime.dotnet.jit.methods_compiled.count, The number of times the JIT compiler compiled a method since the process start. The JIT compiler may be invoked multiple times for the same method to com
pile with different generic parameters, or because tiered compilation requested different optimization settings., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2038076Z, 2023-10-02T03:40:45.2515577Z] LongSum
Value: 12450

Export process.runtime.dotnet.jit.compilation_time, The amount of time the JIT compiler has spent compiling methods since the process start., Unit: ns, Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2038220Z, 2023-10-02T03:40:45.2515581Z] LongSum
Value: 2529930900

Export process.runtime.dotnet.monitor.lock_contention.count, The number of times there was contention when trying to acquire a monitor lock since the process start. Monitor locks are commonly acquired by using t
he lock keyword in C#, or by calling Monitor.Enter() and Monitor.TryEnter()., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2038356Z, 2023-10-02T03:40:45.2515585Z] LongSum
Value: 24

Export process.runtime.dotnet.thread_pool.threads.count, The number of thread pool threads that currently exist., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2038507Z, 2023-10-02T03:40:45.2515588Z] LongSumNonMonotonic
Value: 5

Export process.runtime.dotnet.thread_pool.completed_items.count, The number of work items that have been processed by the thread pool since the process start., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.
0
(2023-10-02T03:39:21.2038641Z, 2023-10-02T03:40:45.2515592Z] LongSum
Value: 85

Export process.runtime.dotnet.thread_pool.queue.length, The number of work items that are currently queued to be processed by the thread pool., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2038766Z, 2023-10-02T03:40:45.2515598Z] LongSumNonMonotonic
Value: 0

Export process.runtime.dotnet.timer.count, The number of timer instances that are currently active. Timers can be created by many sources such as System.Threading.Timer, Task.Delay, or the timeout in a Cancellat
ionSource. An active timer is registered to tick at some point in the future and has not yet been canceled., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2038961Z, 2023-10-02T03:40:45.2515604Z] LongSumNonMonotonic
Value: 1

Export process.runtime.dotnet.assemblies.count, The number of .NET assemblies that are currently loaded., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2039125Z, 2023-10-02T03:40:45.2515608Z] LongSumNonMonotonic
Value: 124

Export process.runtime.dotnet.exceptions.count, Count of exceptions that have been thrown in managed code, since the observation started. The value will be unavailable until an exception has been thrown after Op
enTelemetry.Instrumentation.Runtime initialization., Meter: OpenTelemetry.Instrumentation.Runtime/1.5.1.0
(2023-10-02T03:39:21.2039277Z, 2023-10-02T03:40:45.2515612Z] LongSum
Value: 7

Export process.memory.usage, The amount of physical memory allocated for this process., Unit: By, Meter: OpenTelemetry.Instrumentation.Process/0.5.0.3
(2023-10-02T03:39:21.2039416Z, 2023-10-02T03:40:45.2515618Z] LongSumNonMonotonic
Value: 65191936
```

接著要開始把資料送到ELK，修改剛剛的configure如下:

```
$module_url = "https://github.com/open-telemetry/opentelemetry-dotnet-
instrumentation/releases/latest/download/OpenTelemetry.DotNet.Auto.psm1"
$download_path = Join-Path $env:temp "OpenTelemetry.DotNet.Auto.psm1"
Invoke-WebRequest -Uri $module_url -OutFile $download_path -UseBasicParsing
Import-Module $download_path
Install-OpenTelemetryCore
$env:OTEL_TRACES_EXPORTER="otlp"
$env:OTEL_METRICS_EXPORTER="otlp"
$env:OTEL_LOGS_EXPORTER="otlp"
$env:OTEL_RESOURCE_ATTRIBUTES="service.name=rolling,service.version=1.0,deployment
.environment=production"
$env:OTEL_EXPORTER_OTLP_ENDPOINT="http://localhost:8200"
$env:OTEL_EXPORTER_OTLP_HEADERS="Authorization=Bearer
aExDTkE0c0JPc2prZXVqSkZmbjQ6MWI2dl9rUkdRYkNRR2dfamV3NnFlUQ=="
$env:OTEL_DOTNET_AUTO_TRACES_CONSOLE_EXPORTER_ENABLED="false"
$env:OTEL_DOTNET_AUTO_METRICS_CONSOLE_EXPORTER_ENABLED="false"
$env:OTEL_DOTNET_AUTO_LOGS_CONSOLE_EXPORTER_ENABLED="false"
Register-OpenTelemetryForCurrentSession -OTelServiceName "RollDiceService"
Register-OpenTelemetryForIIS # for IIS
```

`Register-OpenTelemetryForIIS`這行為將Otel註冊到IIS，若不需要可以不用執行。修改完config之後重啟程式應可以看到資料已經送到ELK了。

使用Auto instrumentation 於 現有IIS的專案

針對現有部屬於IIS的專案，可以透過以下步驟進行Auto instrumentation

1. 開啟power-shell (需要管理員權限)，並執行以下指令

~~~
```bash
$module_url = "https://github.com/open-telemetry/opentelemetry-dotnet-instrumentation/releases/latest/download/OpenTelemetry.DotNet.Auto.psm1"
$download_path = Join-Path $env:temp "OpenTelemetry.DotNet.Auto.psm1"
Invoke-WebRequest -Uri $module_url -OutFile $download_path -UseBasicParsing
Import-Module $download_path
Install-OpenTelemetryCore
Register-OpenTelemetryForIIS # for IIS
~~~

2. 修改專案的`web.config`，加入以下內容

```
    <add key="OTEL_TRACES_EXPORTER" value="otlp" />
    <add key="OTEL_METRICS_EXPORTER" value="otlp" />
    <add key="OTEL_LOGS_EXPORTER" value="otlp" />
    <add key="OTEL_RESOURCE_ATTRIBUTES" value="service.name=B2EWebAPI,service.version=1.0,deployment.environment=production" /> <!--需依照需求修改-->
    <add key="OTEL_EXPORTER_OTLP_ENDPOINT" value="http://localhost:8200" /> <!--需依照需求修改-->
    <add key="OTEL_EXPORTER_OTLP_HEADERS" value="Authorization=Bearer aExDTkE0c0JPc2prZXVqSkZmbjQ6MWI2dl9rUkdRYkNRR2dfamV3NnFlUQ==" /> <!--需依照需求修改-->
```

若為.net core的專案則是在web.config內加入下述內容

```
 <environmentVariable name="OTEL_DOTNET_AUTO_TRACES_CONSOLE_EXPORTER_ENABLED" value="true" />
          <environmentVariable name="OTEL_DOTNET_AUTO_METRICS_CONSOLE_EXPORTER_ENABLED" value="true" />
          <environmentVariable name="OTEL_TRACES_EXPORTER" value="otlp" />
          <environmentVariable name="OTEL_METRICS_EXPORTER" value="otlp" />
          <environmentVariable name="OTEL_LOGS_EXPORTER" value="otlp" />
          <environmentVariable name="OTEL_RESOURCE_ATTRIBUTES" value="service.name=rolling,service.version=1.0,deployment.environment=production" />
          <environmentVariable name="OTEL_EXPORTER_OTLP_ENDPOINT" value="http://localhost:8200" />
```

```
          <environmentVariable name="OTEL_EXPORTER_OTLP_HEADERS"
value="Authorization=Bearer
aExDTkE0c0JPc2prZXVqSkZmbjQ6MWI2dl9rUkdRYkNRR2dfamV3NnFlUQ==" />
```

設定完後一樣需要重新建置並重啟IIS，應該就可以看到資料了。

可參考 Instrument an ASP.NET application deployed on IIS

雜記

1. 寫在docker-compose的參數優先於進入container裡面設定的參數，例如`elasticsearch`的`discovery.type=single-node`，在container內設定後，會被`docker-compose.yml`裡面的`environment`覆蓋掉，因此要在`docker-compose.yml`裡面設定。
2. ELK 8.X系列建議都是用Fleet去設定，原有的 beats系列會逐漸退場

Reference

Windows Powershell XXXX.ps1 檔案無法載入

喬叔帶你上手 Elastic Stack - 探索與實踐 Observability 系列

云原生观测性--OpenTelemetry 之实战篇

APM-Server Error talk to ES

APM TSL Error

ElasticsSearch certificate

ElasticsSearch Security Setup

Basic ElasticsSearch Security Setup

ssl

Elastic APM 8.0

Elastic APM 8

Kibana encryption error

OpenTelemetry .NET