使用者指南

AWS 應用程式網格

AWS 應用程式網格: 使用者指南

Table of Contents

什麼是 AWS App Mesh？

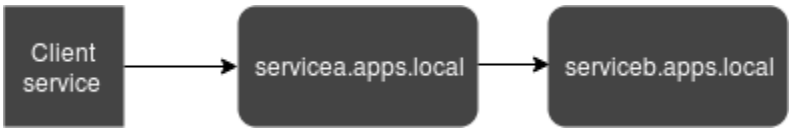
> **Important**
>
> 支援終止通知：在 2026 年 9 月 30 日， AWS 將停止對 的支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS App Mesh 是一種服務網格，可讓您輕鬆監控和控制服務。服務網格是專門處理service-to-service通訊的基礎設施層，通常透過與應用程式程式碼一起部署的一系列輕量型網路代理。App Mesh 將服務通訊方式標準化，為您提供端對端可見性，並有助於確保應用程式的高可用性。App Mesh 對應用程式中的每個服務提供一致的可見性和網路流量控制。

將 App Mesh 新增至範例應用程式

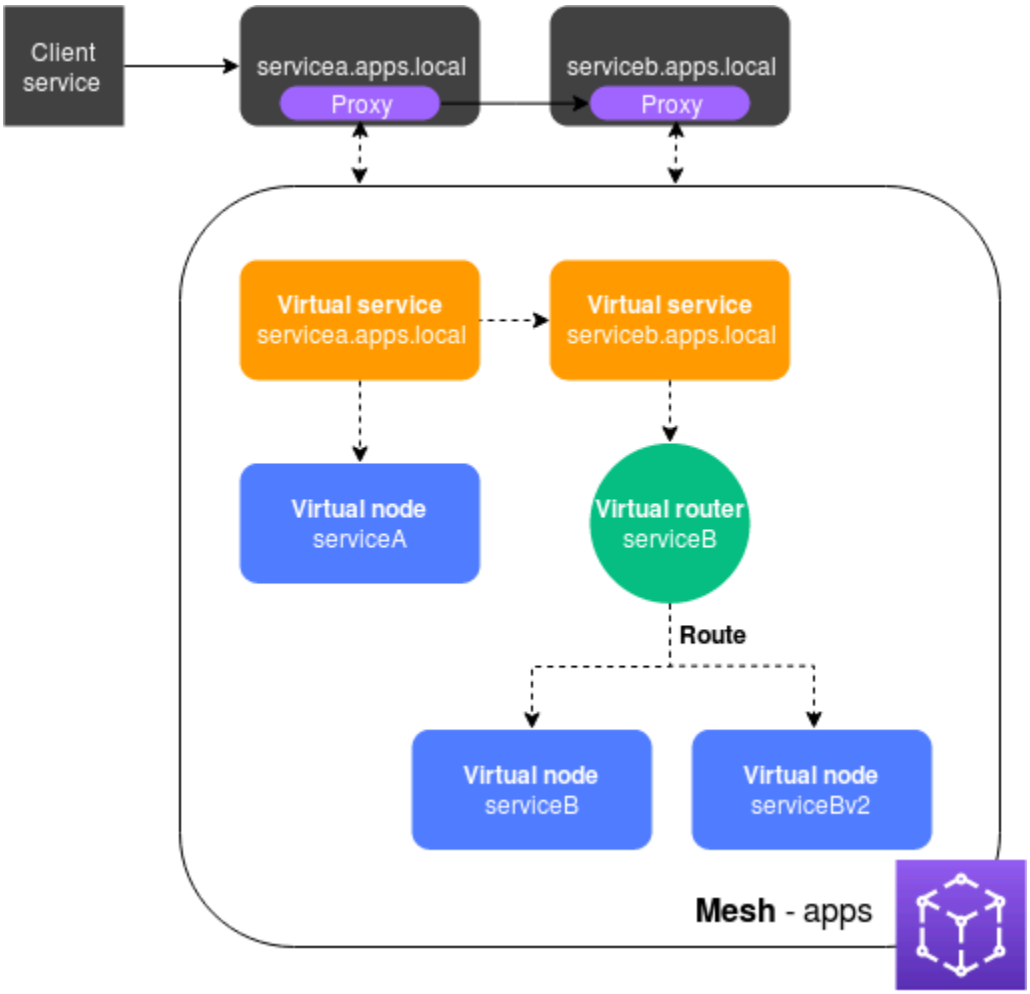
> **Important**
>
> 支援終止通知：在 2026 年 9 月 30 日， AWS 將停止對 的支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

請考慮以下不使用 App Mesh 的簡單範例應用程式。這兩個服務可以在 Amazon Elastic Container Service (Amazon ECS) AWS Fargate、Amazon Elastic Kubernetes Service (Amazon EKS)、Amazon Elastic Compute Cloud (Amazon EC2) 執行個體上的 Kubernetes，或使用 Docker 的 Amazon EC2 執行個體上執行。



在此圖中， `serviceA`和 `serviceB` 都可以透過 `apps.local` 命名空間探索。例如，假設您決定部署`serviceb.apps.local`名為 的新版本`servicebv2.apps.local`。接下來，您想要將一定百分比的流量從 引導`servicea.apps.local`至 `serviceb.apps.local`，並將一定百分比引導至 `servicebv2.apps.local`。當您確定 `servicebv2` 效能良好時，您想要將 100% 的流量傳送至其中。

App Mesh 可協助您執行此操作，而無需變更任何應用程式碼或註冊的服務名稱。如果您搭配此範例應用程式使用 App Mesh，則網格看起來可能會如下圖所示。



在此組態中，服務不會再直接彼此通訊。相反地，它們會透過代理彼此通訊。使用 `servicea.apps.local`服務部署的代理會讀取 App Mesh 組態，並根據組態`servicebv2.apps.local`將流量傳送至 `serviceb.apps.local`或 。

App Mesh 的元件

App Mesh 由下列元件組成，如先前範例所示：

- 服務網格 – 服務網格是位於其中之服務之間的網路流量邏輯界限。在此範例中，網格名為 `apps`，其中包含網格的所有其他資源。如需詳細資訊，請參閱服務網格。
- 虛擬服務 – 虛擬服務是由虛擬節點直接或間接透過虛擬路由器提供的實際服務抽象。在圖中，兩個虛擬服務代表兩個實際服務。虛擬服務的名稱是實際服務的可探索名稱。當虛擬服務和實際服務具有

相同的名稱時，多個服務可以使用在實作 App Mesh 之前所使用的相同名稱彼此通訊。如需詳細資訊，請參閱虛擬服務。

- 虛擬節點 – 虛擬節點可做為可探索服務的邏輯指標，例如 Amazon ECS 或 Kubernetes 服務。對於每個虛擬服務，您至少會有一個虛擬節點。在圖中，servicea.apps.local虛擬服務會取得名為之虛擬節點的組態資訊serviceA。serviceA 虛擬節點是以服務探索servicea.apps.local的名稱設定。serviceb.apps.local 虛擬服務設定為透過名為 的serviceBv2虛擬路由器將流量路由到 serviceB和虛擬節點serviceB。如需詳細資訊，請參閱虛擬節點。
- 虛擬路由器和路由 – 虛擬路由器會處理網格中一或多個虛擬服務的流量。路由會與虛擬路由器相關聯。路由用於比對虛擬路由器的請求，並將流量分配到其相關聯的虛擬節點。在上圖中，serviceB虛擬路由器的路由會將一定百分比的流量導向虛擬serviceB節點，以及將一定百分比的流量導向serviceBv2虛擬節點。您可以設定路由到特定虛擬節點的流量百分比，並隨時間變更。您可以根據 HTTP 標頭、URL 路徑或 gRPC 服務和方法名稱等條件路由流量。如果回應中發生錯誤，您可以設定重試政策以重試連線。例如，在圖中，如果 serviceb.apps.local傳回特定類型的錯誤，則路由的重試政策可以指定serviceb.apps.local對 的連線重試五次，重試嘗試之間間隔十秒。如需詳細資訊，請參閱虛擬路由器及路由。
- Proxy – 您可以在建立網格及其資源後，將服務設定為使用 Proxy。代理會讀取 App Mesh 組態，並適當地引導流量。在圖中，從 servicea.apps.local 到 的所有通訊serviceb.apps.local都會經過與每個服務一起部署的代理。服務會使用與導入 App Mesh 之前相同的服務探索名稱彼此通訊。由於代理會讀取 App Mesh 組態，因此您可以控制兩個 服務如何互相通訊。當您想要變更 App Mesh 組態時，您不需要自行或代理變更或重新部署服務。如需詳細資訊，請參閱Envoy 影像。

如何開始

若要使用 App Mesh，您必須擁有執行於 的現有服務 AWS Fargate、Amazon ECS、Amazon EKS、Amazon EC2 上的 Kubernetes 或 Amazon EC2 搭配 Docker。

若要開始使用 App Mesh，請參閱下列其中一個指南：

- App Mesh 和 Amazon ECS 入門
- App Mesh 和 Kubernetes 入門
- App Mesh 和 Amazon EC2 入門

存取 App Mesh

您可以透過下列方式使用 App Mesh：

AWS Management Console

主控台是瀏覽器型界面，可用來管理 App Mesh 資源。您可以在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。

AWS CLI

為廣泛的 AWS 產品提供命令，並支援 Windows、Mac 和 Linux。若要開始使用，請參閱《AWS Command Line Interface 使用者指南》。如需 App Mesh 命令的詳細資訊，請參閱《 AWS CLI 命令參考》中的 appmesh。

AWS Tools for Windows PowerShell

為在 PowerShell 環境中編寫指令碼的人員提供廣泛的 AWS 產品命令。若要開始使用，請參閱《AWS Tools for Windows PowerShell 使用者指南》。如需有關 App Mesh 的 cmdlet 的詳細資訊，請參閱AWS 《 Tools for PowerShell Cmdlet Reference》中的 App Mesh。

AWS CloudFormation

可讓您建立範本，描述您想要的所有 AWS 資源。使用 範本，為您 AWS CloudFormation 佈建和設定 資源。若要開始使用，請參閱《AWS CloudFormation 使用者指南》。如需 App Mesh 資源類型的詳細資訊，請參閱 範本參考中的 App Mesh 資源類型AWS CloudFormation 參考。

AWS SDKs

我們也提供 SDKs，可讓您從各種程式設計語言存取 App Mesh。SDKs會自動處理下列任務：

- 加密簽署服務請求
- 重試請求
- 處理錯誤回應

如需可用 SDK 的詳細資訊，請參閱 Amazon Web Services 適用工具。

如需 App Mesh APIs的詳細資訊，請參閱 AWS App Mesh API 參考。

App Mesh 入門

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

您可以搭配部署至 Amazon ECS、Kubernetes （部署至您自己的 Amazon EC2 執行個體或在 Amazon EKS 上執行） 和 Amazon EC2 的應用程式使用 App Mesh。若要開始使用 App Mesh，請選取您要搭配 App Mesh 使用之應用程式部署到其中的其中一項服務。在完成其中一個入門指南後，您可以隨時啟用其他 服務中的應用程式，以使用 App Mesh。

主題

- AWS App Mesh 和 Amazon ECS 入門
- AWS App Mesh 和 Kubernetes 入門
- AWS App Mesh 和 Amazon EC2 入門
- App Mesh 範例

AWS App Mesh 和 Amazon ECS 入門

> **Important**
>
> 終止支援通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題可協助您 AWS App Mesh 在 Amazon ECS 上執行的實際服務使用 。本教學課程涵蓋數種 App Mesh 資源類型的基本功能。

案例

若要說明如何使用 App Mesh，假設您擁有具有下列特性的應用程式：

- 包含兩個名為 serviceA和 的服務serviceB。
- 這兩個服務都註冊到名稱為 apps.local 的命名空間。
- ServiceA 與 serviceB 透過 HTTP/2，連接埠 80 進行通訊。
- 您已部署 serviceB 第 2 版，並在 apps.local 命名空間中將其註冊為名稱 serviceBv2。

您有以下要求：

- 您想要將 75% 的流量從 serviceA 傳送到 serviceB，並將 25% 的流量傳送到 serviceBv2。透過僅將 25% 傳送到 serviceBv2，您可以在從 傳送 100% 的流量之前，驗證它是否沒有錯誤serviceA。
- 您希望能夠輕鬆地調整流量權重，以便證實流量可靠之後，能夠 100% 流入 serviceBv2。將所有流量傳送到 後serviceBv2，您想要停止 serviceB。
- 您不想變更實際服務的任何現有應用程式碼或服務探索註冊，以符合先前的要求。

為了滿足您的需求，您決定建立 App Mesh 服務網格，其中包含虛擬服務、虛擬節點、虛擬路由器和路由。實作網格後，您會更新您的服務以使用 Envoy 代理。一旦更新，您的服務會透過 Envoy 代理彼此通訊，而非直接相互通訊。

先決條件

> **Important**
>
> 終止支援通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

- 對 App Mesh 概念的現有理解。如需詳細資訊，請參閱什麼是 AWS App Mesh？。
- 對 Amazon ECSs概念的現有理解。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的什麼是 Amazon ECS。
- App Mesh 支援已向 DNS 註冊的 Linux 服務 AWS Cloud Map，或兩者皆支援。若要使用此入門指南，建議您擁有三個已向 DNS 註冊的現有服務。本主題中的程序假設現有服務名為 serviceA、和 serviceB，serviceBv2並且所有服務都可以透過名為 的命名空間探索apps.local。

  即使服務不存在，您也可以建立服務網格及其資源，但在部署實際服務之前，您無法使用網格。如需 Amazon ECS 上服務探索的詳細資訊，請參閱服務探索。若要使用服務探索建立 Amazon ECS 服

務，請參閱教學課程：使用服務探索建立服務。如果您還沒有執行中的服務，則可以使用服務探索建立 Amazon ECS 服務。

步驟 1：建立網格和虛擬服務

服務網格是在它之內各服務之間網路流量的邏輯邊界。如需詳細資訊，請參閱服務網格。虛擬服務是實際服務的抽象化。如需詳細資訊，請參閱虛擬服務。

建立下列資源：

- 名稱為 apps 的網格，因為案例中的所有服務皆註冊到 apps.local 命名空間。
- 名稱為 serviceb.apps.local 的虛擬服務，因為虛擬服務代表可使用該名稱探索的服務，而且您不想將程式碼變更為參照其他名稱。稍後的步驟會新增名稱為 servicea.apps.local 的虛擬服務。

您可以使用 AWS Management Console 或 1.18.116 AWS CLI 版或更新版本，或 2.0.38 版或更新版本來完成下列步驟。如果使用 AWS CLI，請使用 aws --version命令來檢查已安裝的 AWS CLI 版本。如果您沒有安裝 1.18.116 或更新版本或 2.0.38 或更新版本，則必須安裝或更新 AWS CLI。為您要使用的工具選取索引標籤。

AWS Management Console

1. 在 https://console.aws.amazon.com/appmesh/get-started：// App Mesh 主控台初次執行精靈。
2. 對於 Mesh name (網格名稱)，輸入 **apps**。
3. 對於 Virtual service name (虛擬服務名稱)，輸入 **serviceb.apps.local**。
4. 若要繼續，請選擇 Next (下一步)。

AWS CLI

1. 使用 create-mesh 命令建立網格。

```
aws appmesh create-mesh --mesh-name apps
```

2. 使用 create-virtual-service 命令建立虛擬服務。

```
aws appmesh create-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local --spec {}
```

步驟 2：建立虛擬節點

虛擬節點可做為實際服務的邏輯指標。如需詳細資訊，請參閱虛擬節點。

建立名稱為 serviceB 的虛擬節點，因為其中一個虛擬節點代表名稱為 serviceB 的實際服務。虛擬節點代表的實際服務可透過 DNS (主機名稱為 serviceb.apps.local) 探索。或者，您可以使用 探索實際的服務 AWS Cloud Map。虛擬節點將會在連接埠 80 上使用 HTTP/2 通訊協定來接聽流量。也支援其他通訊協定，以及運作狀態檢查。您將在稍後的步驟中為 serviceA 和 serviceBv2 建立虛擬節點。

AWS Management Console

1. 對於 Virtual node name (虛擬節點名稱)，輸入 **serviceB**。
2. 針對服務探索方法，選擇 DNS，然後輸入 **serviceb.apps.local** DNS 主機名稱。
3. 在接聽程式組態下，選擇通訊協定的 http2，然後輸入連接埠**80**的 。
4. 若要繼續，請選擇 Next (下一步)。

AWS CLI

1. 使用下列內容建立名為 create-virtual-node-serviceb.json 的檔案：

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceB.apps.local"
```

```
            }
        }
    },
    "virtualNodeName": "serviceB"
}
```

2. 使用 JSON 檔案做為輸入，搭配 create-virtual-node 命令建立虛擬節點。

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-serviceb.json
```

步驟 3：建立虛擬路由器和路由

虛擬路由器會路由網格內一或多個虛擬服務的流量。如需詳細資訊，請參閱虛擬路由器及路由。

建立下列資源：

- 名為 `serviceB` 的虛擬路由器，因為 `serviceB.apps.local` 虛擬服務不會啟動與任何其他服務的對外通訊。請記住，您先前建立的虛擬服務是實際 `serviceb.apps.local` 服務的抽象。虛擬服務會將流量傳送至虛擬路由器。虛擬路由器會使用連接埠 80 上的 HTTP/2 通訊協定接聽流量。也支援其他通訊協定。
- 名為 `serviceB` 的路由。它將 100% 的流量路由到`serviceB`虛擬節點。新增`serviceBv2`虛擬節點後，權重會進入後續步驟。雖然未涵蓋在本指南中，但您可以為路由新增其他篩選條件，並新增重試政策，使 Envoy 代理在遇到通訊問題時多次嘗試將流量傳送至虛擬節點。

AWS Management Console

1. 對於 Virtual router name (虛擬路由器名稱)，輸入 **`serviceB`**。
2. 在接聽程式組態下，為通訊協定選擇 http2，並為連接埠指定 **`80`** 。
3. 對於 Route name (路由名稱)，輸入 **`serviceB`**。
4. 對於 Route type (路由類型)，選擇 http2。
5. 針對目標組態下的虛擬節點名稱，選取 `serviceB`並輸入 **`100`**表示權重。
6. 在相符組態下，選擇方法。
7. 若要繼續，請選擇 Next (下一步)。

AWS CLI

1. 建立虛擬路由器。

   a. 使用下列內容建立名為 `create-virtual-router.json` 的檔案：

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ]
    },
    "virtualRouterName": "serviceB"
}
```

   b. 使用 JSON 檔案做為輸入，搭配 create-virtual-router 命令建立虛擬路由器。

```
aws appmesh create-virtual-router --cli-input-json file://create-virtual-
router.json
```

2. 建立路由。

   a. 使用下列內容建立名為 `create-route.json` 的檔案：

```
{
    "meshName" : "apps",
    "routeName" : "serviceB",
    "spec" : {
        "httpRoute" : {
            "action" : {
                "weightedTargets" : [
                    {
                        "virtualNode" : "serviceB",
                        "weight" : 100
                    }
                ]
            },
```

```
            "match" : {
                "prefix" : "/"
            }
        }
    },
    "virtualRouterName" : "serviceB"
}
```

b. 使用 JSON 做為輸入，搭配 create-route 命令建立路由。

```
aws appmesh create-route --cli-input-json file://create-route.json
```

步驟 4：檢閱和建立

依照先前的指示檢閱設定。

AWS Management Console

如果您需要在任何區段中進行變更，請選擇 Edit (編輯)。對這些設定感到滿意後，請選擇 Create mesh (建立網格)。

Status (狀態) 畫面會顯示所有已建立的網格資源。選取 View mesh (檢視網格) 即可在主控台中檢視建立的資源。

AWS CLI

檢閱您使用 describe-mesh 命令建立的網格設定。

```
aws appmesh describe-mesh --mesh-name apps
```

檢閱您使用 describe-virtual-service 命令建立的虛擬服務設定。

```
aws appmesh describe-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local
```

檢閱您使用 describe-virtual-node 命令建立的虛擬節點設定。

```
aws appmesh describe-virtual-node --mesh-name apps --virtual-node-name serviceB
```

檢閱您使用 describe-virtual-router 命令建立的虛擬路由器設定。

```
aws appmesh describe-virtual-router --mesh-name apps --virtual-router-name serviceB
```

檢閱您使用 describe-route 命令建立的路由設定。

```
aws appmesh describe-route --mesh-name apps \
    --virtual-router-name serviceB  --route-name serviceB
```

步驟 5：建立其他資源

若要完成案例，您必須：

- 建立一個名為 serviceBv2 的虛擬節點，以及另一個名為 serviceA 的虛擬節點。這兩個虛擬節點都會透過 HTTP/2 連接埠 80 接聽請求。針對serviceA虛擬節點，設定 的後端serviceb.apps.local。來自serviceA虛擬節點的所有傳出流量都會傳送至名為 的虛擬服務serviceb.apps.local。雖然未涵蓋在本指南中，但您也可以指定將虛擬節點的存取日誌寫入其中的檔案路徑。
- 建立名為 的額外虛擬服務servicea.apps.local，將所有流量直接傳送至serviceA虛擬節點。
- 更新您在上一個步驟中建立的 serviceB 路由，將 75% 的流量傳送到 serviceB 虛擬節點，以及 25% 的流量傳送到 serviceBv2 虛擬節點。隨著時間的推移，您可以繼續修改權重，直到 serviceBv2 收到 100% 的流量為止。將所有流量傳送到 後serviceBv2，您可以關閉並停止serviceB虛擬節點和實際服務。當您更改權重時，您的程式碼不需要任何修改，因為 serviceb.apps.local 虛擬和實際服務名稱不會變更。記住，serviceb.apps.local 虛擬服務會將流量傳送至虛擬路由器，接著虛擬路由器會將流量路由至虛擬節點。虛擬節點的服務探索名稱可以隨時變更。

AWS Management Console

1. 在左側導覽窗格中，選取 Meshes (網格)。
2. 選取您在上一個步驟中建立的 apps 網格。
3. 在左側導覽窗格中，選取 Virtual nodes (虛擬節點)。
4. 選擇 Create virtual node (建立虛擬節點)。
5. 對於 Virtual node name (虛擬節點名稱) 輸入 **serviceBv2**、對於 Service discovery method (服務探索方法) 選擇 DNS，然後對於 DNS hostname (DNS 主機名稱) 輸入 **servicebv2.apps.local**。
6. 針對接聽程式組態，選取通訊協定的 http2，然後針對連接埠輸入 **80** 。

7. 選擇 Create virtual node (建立虛擬節點)。
8. 再次選擇 Create virtual node (建立虛擬節點)。輸入 **serviceA** 做為虛擬節點名稱。對於 Service discovery method (服務探索方法)，選擇 DNS，並針對 DNS hostname (DNS 主機名稱) 輸入 **servicea.apps.local**。
9. 對於在新增後端下輸入虛擬服務名稱，輸入 **serviceb.apps.local**。
10. 在接聽程式組態下，選擇通訊協定的 http2，在連接埠**80**中輸入 ，然後選擇建立虛擬節點。
11. 在左側導覽窗格中，選取 Virtual routers (虛擬路由器)，然後從清單中選取 serviceB 虛擬路由器。
12. 在 Routes (路由) 下方，選取您在上一個步驟中建立的路由 (名為 ServiceB)，然後選擇 Edit (編輯)。
13. 在目標、虛擬節點名稱下，將 的權重值變更為 serviceB **75**。
14. 選擇新增目標，serviceBv2從下拉式清單中選擇，並將權重的值設定為 **25**。
15. 選擇儲存。
16. 在左側瀏覽窗格中，選取 Virtual services (虛擬服務)，然後選擇 Create virtual service (建立虛擬服務)。
17. **servicea.apps.local** 針對虛擬服務名稱輸入 ，針對提供者選取虛擬節點，serviceA針對虛擬節點選取 ，然後選擇建立虛擬服務。

AWS CLI

1. 建立 serviceBv2 虛擬節點。

   a. 使用下列內容建立名為 create-virtual-node-servicebv2.json 的檔案：

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
```

```
                "hostname": "serviceBv2.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceBv2"
}
```

b. 建立虛擬節點。

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicebv2.json
```

2. 建立 serviceA 虛擬節點。

a. 使用下列內容建立名為 create-virtual-node-servicea.json 的檔案：

```
{
   "meshName" : "apps",
   "spec" : {
      "backends" : [
         {
            "virtualService" : {
               "virtualServiceName" : "serviceb.apps.local"
            }
         }
      ],
      "listeners" : [
         {
            "portMapping" : {
               "port" : 80,
               "protocol" : "http2"
            }
         }
      ],
      "serviceDiscovery" : {
         "dns" : {
            "hostname" : "servicea.apps.local"
         }
      }
   },
   "virtualNodeName" : "serviceA"
}
```

b. 建立虛擬節點。

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicea.json
```

3. 更新您在上一個步驟中建立的 serviceb.apps.local 虛擬服務，將其流量傳送至 serviceB 虛擬路由器。最初建立虛擬服務時，它不會傳送流量到任何地方，因為 serviceB 虛擬路由器尚未建立。

a. 使用下列內容建立名為 update-virtual-service.json 的檔案：

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualRouter" : {
            "virtualRouterName" : "serviceB"
         }
      }
   },
   "virtualServiceName" : "serviceb.apps.local"
}
```

b. 使用 update-virtual-service 命令更新虛擬服務。

```
aws appmesh update-virtual-service --cli-input-json file://update-virtual-service.json
```

4. 更新您在上一個步驟中建立的 serviceB 路由。

a. 使用下列內容建立名為 update-route.json 的檔案：

```
{
   "meshName" : "apps",
   "routeName" : "serviceB",
   "spec" : {
      "http2Route" : {
         "action" : {
            "weightedTargets" : [
               {
                  "virtualNode" : "serviceB",
                  "weight" : 75
```

```
                },
                {
                   "virtualNode" : "serviceBv2",
                   "weight" : 25
                }
             ]
          },
          "match" : {
             "prefix" : "/"
          }
       }
   },
   "virtualRouterName" : "serviceB"
}
```

b. 使用 update-route 命令更新路由。

```
aws appmesh update-route --cli-input-json file://update-route.json
```

5. 建立 serviceA 虛擬服務。

a. 使用下列內容建立名為 create-virtual-servicea.json 的檔案：

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualNode" : {
            "virtualNodeName" : "serviceA"
         }
      }
   },
   "virtualServiceName" : "servicea.apps.local"
}
```

b. 建立虛擬服務。

```
aws appmesh create-virtual-service --cli-input-json file://create-virtual-
servicea.json
```

網格摘要

在您建立服務網格之前，您有三個名稱為 servicea.apps.local、serviceb.apps.local 和 servicebv2.apps.local 的實際服務。除了實際服務以外，您目前擁有包含代表實際服務之下列資源的服務網格：

- 兩個虛擬服務。代理會透過虛擬路由器將 servicea.apps.local 虛擬服務的所有流量傳送至 serviceb.apps.local 虛擬服務。
- 名稱為 serviceA、serviceB 和 serviceBv2 的三個虛擬節點。Envoy 代理會使用針對虛擬節點設定的服務探索資訊，來查詢實際服務的 IP 地址。
- 具有單一路由的虛擬路由器，其指示 Envoy 代理將 75% 的傳入流量路由到 serviceB 虛擬節點，將 25% 的流量路由到 serviceBv2 虛擬節點。

步驟 6：更新服務

在建立網格之後，您需要完成下列任務：

- 授權您使用每個 Amazon ECS 任務部署的 Envoy 代理程式，以讀取一或多個虛擬節點的組態。如需如何授權代理的詳細資訊，請參閱代理授權。
- 更新每個現有的 Amazon ECS 任務定義，以使用 Envoy 代理。

登入資料

Envoy 容器需要 AWS Identity and Access Management 登入資料，才能簽署傳送到 App Mesh 服務的請求。對於使用 Amazon EC2 啟動類型部署的 Amazon ECS 任務，憑證可以來自執行個體角色或任務 IAM 角色。在 Linux 容器上使用 Fargate 部署的 Amazon ECS 任務無法存取提供執行個體 IAM 設定檔憑證的 Amazon EC2 中繼資料伺服器。若要提供登入資料，您必須將 IAM 任務角色連接至使用 Linux 上的 Fargate 容器類型部署的任何任務。

如果使用 Amazon EC2 啟動類型部署任務，且 Amazon EC2 中繼資料伺服器遭到封鎖存取，如任務 IAM 角色中的重要註釋中所述，則任務 IAM 角色也必須連接至任務。您指派給執行個體或任務的角色必須連接 IAM 政策，如 Proxy 授權中所述。

使用 更新您的任務定義 AWS CLI

您可以使用 Amazon ECS AWS CLI 命令 register-task-definition。以下任務定義範例說明如何為您的服務設定 App Mesh。

> **Note**
>
> 透過主控台設定 Amazon ECS 的 App Mesh 無法使用。

任務定義 json

代理組態

若要將 Amazon ECS 服務設定為使用 App Mesh，您服務的任務定義必須具有下列代理組態區段。將代理組態 `type` 設為 `APPMESH`，將 `containerName` 設為 `envoy`。相應地設定下列屬性值。

`IgnoredUID`

對於使用此使用者 ID 的程序，Envoy 代理不會路由來自這些程序的流量。您可以為此屬性值選擇您想要的任何使用者 ID，但此 ID 必須與任務定義中 Envoy 容器的 `user` ID 相同。這配對可讓 Envoy 忽略自己的流量，而不使用代理。我們的範例是基於歷史用途使用 *`1337`*。

`ProxyIngressPort`

這是 Envoy 代理容器的傳入連接埠。將此值設為 `15000`。

`ProxyEgressPort`

這是 Envoy 代理容器的傳出連接埠。將此值設為 `15001`。

`AppPorts`

指定應用程式容器接聽的任何傳入連接埠。在這個範例中，應用程式容器會接聽連接埠 *`9080`*。您指定的連接埠必須符合在虛擬節點接聽程式上設定的連接埠。

`EgressIgnoredIPs`

Envoy 不會代理將流量送往這些 IP 地址。將此值設定為 `169.254.170.2,169.254.169.254`，這會忽略 Amazon EC2 中繼資料伺服器和 Amazon ECS 任務中繼資料端點。中繼資料端點提供任務登入資料的 IAM 角色。您可以新增其他地址。

`EgressIgnoredPorts`

您可以新增以逗號分隔的連接埠清單。Envoy 不會代理將流量送往這些連接埠。即使您沒有列出連接埠，連接埠 22 也會被忽略。

> **Note**
>
> 可忽略的傳出連接埠數目上限為 15。

```
"proxyConfiguration": {
 "type": "APPMESH",
 "containerName": "envoy",
 "properties": [{
   "name": "IgnoredUID",
   "value": "1337"
  },
  {
   "name": "ProxyIngressPort",
   "value": "15000"
  },
  {
   "name": "ProxyEgressPort",
   "value": "15001"
  },
  {
   "name": "AppPorts",
   "value": "9080"
  },
  {
   "name": "EgressIgnoredIPs",
   "value": "169.254.170.2,169.254.169.254"
  },
  {
   "name": "EgressIgnoredPorts",
   "value": "22"
  }
 ]
}
```

應用程式容器 Envoy 相依性

任務定義中的應用程式容器必須等待 Envoy 代理引導並啟動之後才能啟動。為了確保發生這種情況，您可以在每個應用程式容器定義中設定dependsOn區段，以等待 Envoy 容器報告為 HEALTHY。以下程式碼顯示具有此相依性的應用程式容器定義範例。以下範例中的所有屬性都是必要的。某些屬性值也是必要的，但有些是####。

```
{
 "name": "appName",
 "image": "appImage",
 "portMappings": [{
  "containerPort": 9080,
  "hostPort": 9080,
  "protocol": "tcp"
 }],
 "essential": true,
 "dependsOn": [{
  "containerName": "envoy",
  "condition": "HEALTHY"
 }]
}
```

Envoy 容器定義

您的 Amazon ECS 任務定義必須包含 App Mesh Envoy 容器映像。

所有支援的區域都可以將###取代為 me-south-1、ap-east-1、ap-southeast-3、eu-south-1、 il-central-1和 以外的任何區域af-south-1。

標準

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

符合 FIPS 規範

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod-fips
```

me-south-1

標準

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

ap-east-1

標準

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

ap-southeast-3

標準

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

eu-south-1

標準

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

il-central-1

標準

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

af-south-1

標準

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

Public repository

標準

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.12.1-prod
```

符合 FIPS 規範

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.12.1-prod-fips
```

> **Important**
>
> 僅支援 v1.9.0.0 版或更新版本搭配 App Mesh 使用。

您必須使用 App Mesh Envoy 容器映像，直到 Envoy 專案團隊合併支援 App Mesh 的變更。如需其他詳細資訊，請參閱 GitHub 藍圖問題。

以下範例中的所有屬性都是必要的。某些屬性值也是必要的，但有些是*####*。

> **Note**
>
> - Envoy 容器定義必須標示為 `essential`。
> - 我們建議將 512 CPU 單位和至少 64 MiB 的記憶體配置給 Envoy 容器。在 Fargate 上，您可以設定的最低記憶體為 1024 MiB。
> - Amazon ECS 服務的虛擬節點名稱必須設定為 APPMESH_RESOURCE_ARN 屬性的值。此屬性需要版本 `1.15.0`或更新版本的 Envoy 映像。如需詳細資訊，請參閱Envoy。
> - `user` 設定的值必須與任務定義代理組態中的 `IgnoredUID` 值相符。在此範例中，我們使用 *1337*.
> - 此處顯示的運作狀態檢查會等待 Envoy 容器正確引導，然後再向 Amazon ECS 報告 Envoy 容器運作狀態良好，並準備好啟動應用程式容器。
> - 根據預設，當 Envoy 在指標和追蹤方面參照本身時，App Mesh 會使用您在 APPMESH_RESOURCE_ARN 中指定的資源名稱。您可以藉由使用自己的名稱設定 APPMESH_RESOURCE_CLUSTER 環境變數，以覆寫此行為。此屬性需要版本 `1.15.0`或更新版本的 Envoy 映像。如需詳細資訊，請參閱Envoy。

下列程式碼顯示 Envoy 容器定義範例。

```
{
 "name": "envoy",
 "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-
prod",
 "essential": true,
 "environment": [{
  "name": "APPMESH_RESOURCE_ARN",
  "value": "arn:aws:appmesh:us-west-2:111122223333:mesh/apps/virtualNode/serviceB"
 }],
 "healthCheck": {
  "command": [
   "CMD-SHELL",
   "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
  ],
  "startPeriod": 10,
```

```
  "interval": 5,
  "timeout": 2,
  "retries": 3
 },
 "user": "1337"
}
```

任務定義範例

下列範例 Amazon ECS 任務定義示範如何將上述範例合併為 的任務定義taskB。提供範例，用於建立兩個 Amazon ECS 啟動類型的任務，無論是否使用 AWS X-Ray。視需要變更###值，以從案例中建立名為 taskBv2 和 taskA 的任務定義。以您的網格名稱和虛擬節點名稱代替 APPMESH_RESOURCE_ARN 值，以您的應用程式監聽的連接埠清單代替代理組態 AppPorts 值。根據預設，當 Envoy 在指標和追蹤方面參照本身時，App Mesh 會使用您在 APPMESH_RESOURCE_ARN 中指定的資源名稱。您可以藉由使用自己的名稱設定 APPMESH_RESOURCE_CLUSTER 環境變數，以覆寫此行為。下列範例中的所有屬性都是必要的。某些屬性值也是必要的，但有些是####。

如果您正在執行登入資料區段中所述的 Amazon ECS 任務，則需要將現有的任務 IAM 角色新增至範例。

> **Important**
>
> Fargate 必須使用大於 1024 的連接埠值。

Example Amazon ECS 任務定義的 JSON - Linux 容器上的 Fargate

```
{

   "family" : "taskB",
   "memory" : "1024",
   "cpu" : "0.5 vCPU",
   "proxyConfiguration" : {
      "containerName" : "envoy",
      "properties" : [
         {
            "name" : "ProxyIngressPort",
            "value" : "15000"
         },
         {
            "name" : "AppPorts",
            "value" : "9080"
```

```
         },
         {
            "name" : "EgressIgnoredIPs",
            "value" : "169.254.170.2,169.254.169.254"
         },
         {
            "name": "EgressIgnoredPorts",
            "value": "22"
         },
         {
            "name" : "IgnoredUID",
            "value" : "1337"
         },
         {
            "name" : "ProxyEgressPort",
            "value" : "15001"
         }
      ],
      "type" : "APPMESH"
   },
   "containerDefinitions" : [
      {
         "name" : "appName",
         "image" : "appImage",
         "portMappings" : [
            {
               "containerPort" : 9080,
               "protocol" : "tcp"
            }
         ],
         "essential" : true,
         "dependsOn" : [
            {
               "containerName" : "envoy",
               "condition" : "HEALTHY"
            }
         ]
      },
      {
         "name" : "envoy",
         "image" : "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.29.12.1-prod",
         "essential" : true,
         "environment" : [
```

```
            {
               "name" : "APPMESH_VIRTUAL_NODE_NAME",
               "value" : "mesh/apps/virtualNode/serviceB"
            }
         ],
         "healthCheck" : {
            "command" : [
               "CMD-SHELL",
               "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
            ],
            "interval" : 5,
            "retries" : 3,
            "startPeriod" : 10,
            "timeout" : 2
         },
         "memory" : 500,
         "user" : "1337"
      }
   ],
   "requiresCompatibilities" : [ "FARGATE" ],
   "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
   "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
   "networkMode" : "awsvpc"
}
```

Example JSON for Amazon ECS 任務定義 AWS X-Ray - Linux 容器上的 Fargate

X-Ray 可讓您收集應用程式提供的請求相關資料，並提供可用來視覺化流量的工具。使用適用於 Envoy 的 X-Ray 驅動程式可讓 Envoy 向 X-Ray 報告追蹤資訊。您可以使用 Envoy 組態啟用 X-Ray 追蹤。根據組態，Envoy 會將追蹤資料傳送至做為附屬容器執行的 X-Ray 協助程式，而協助程式會將追蹤轉送至 X-Ray 服務。將追蹤發佈至 X-Ray 之後，您可以使用 X-Ray 主控台來視覺化服務呼叫圖表並請求追蹤詳細資訊。下列 JSON 代表啟用 X-Ray 整合的任務定義。

```
{


   "family" : "taskB",
   "memory" : "1024",
   "cpu" : "512",
   "proxyConfiguration" : {
      "containerName" : "envoy",
      "properties" : [
         {
```

```
            "name" : "ProxyIngressPort",
            "value" : "15000"
         },
         {
            "name" : "AppPorts",
            "value" : "9080"
         },
         {
            "name" : "EgressIgnoredIPs",
            "value" : "169.254.170.2,169.254.169.254"
         },
         {
            "name": "EgressIgnoredPorts",
            "value": "22"
         },
         {
            "name" : "IgnoredUID",
            "value" : "1337"
         },
         {
            "name" : "ProxyEgressPort",
            "value" : "15001"
         }
      ],
      "type" : "APPMESH"
   },
   "containerDefinitions" : [
      {
         "name" : "appName",
         "image" : "appImage",
         "portMappings" : [
            {
               "containerPort" : 9080,
               "protocol" : "tcp"
            }
         ],
         "essential" : true,
         "dependsOn" : [
            {
               "containerName" : "envoy",
               "condition" : "HEALTHY"
            }
         ]
      },
```

```
      {

         "name" : "envoy",
         "image" : "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.29.12.1-prod",
         "essential" : true,
         "environment" : [
            {
               "name" : "APPMESH_VIRTUAL_NODE_NAME",
               "value" : "mesh/apps/virtualNode/serviceB"
            },
            {
               "name": "ENABLE_ENVOY_XRAY_TRACING",
               "value": "1"
            }
         ],
         "healthCheck" : {
            "command" : [
               "CMD-SHELL",
               "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
            ],
            "interval" : 5,
            "retries" : 3,
            "startPeriod" : 10,
            "timeout" : 2
         },
         "memory" : 500,
         "user" : "1337"
      },
      {
         "name" : "xray-daemon",
         "image" : "amazon/aws-xray-daemon",
         "user" : "1337",
         "essential" : true,
         "cpu" : "32",
         "memoryReservation" : "256",
         "portMappings" : [
            {
               "containerPort" : 2000,
               "protocol" : "udp"
            }
         ]
      }
   ],
```

```
    "requiresCompatibilities" : [ "FARGATE" ],
    "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
    "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
    "networkMode" : "awsvpc"
}
```

Example Amazon ECS 任務定義的 JSON - EC2 啟動類型

```
{
  "family": "taskB",
  "memory": "256",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "envoy",
    "properties": [
      {
        "name": "IgnoredUID",
        "value": "1337"
      },
      {
        "name": "ProxyIngressPort",
        "value": "15000"
      },
      {
        "name": "ProxyEgressPort",
        "value": "15001"
      },
      {
        "name": "AppPorts",
        "value": "9080"
      },
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      },
      {
        "name": "EgressIgnoredPorts",
        "value": "22"
      }
    ]
  },
  "containerDefinitions": [
    {
```

```
      "name": "appName",
      "image": "appImage",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.29.12.1-prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/apps/virtualNode/serviceB"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    }
  ],
  "requiresCompatibilities" : [ "EC2" ],
  "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
  "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
```

```
  "networkMode": "awsvpc"
}
```

Example JSON for Amazon ECS 任務定義搭配 AWS X-Ray EC2 啟動類型

```
{
  "family": "taskB",
  "memory": "256",
   "cpu" : "1024",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "envoy",
    "properties": [
      {
        "name": "IgnoredUID",
        "value": "1337"
      },
      {
        "name": "ProxyIngressPort",
        "value": "15000"
      },
      {
        "name": "ProxyEgressPort",
        "value": "15001"
      },
      {
        "name": "AppPorts",
        "value": "9080"
      },
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      },
      {
        "name": "EgressIgnoredPorts",
        "value": "22"
      }
    ]
  },
  "containerDefinitions": [
    {
      "name": "appName",
      "image": "appImage",
```

```
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.29.12.1-prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/apps/virtualNode/serviceB"
        },
        {
         "name": "ENABLE_ENVOY_XRAY_TRACING",
         "value": "1"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    },
    {
      "name": "xray-daemon",
```

```
      "image": "amazon/aws-xray-daemon",
      "user": "1337",
      "essential": true,
      "cpu": 32,
      "memoryReservation": 256,
      "portMappings": [
        {
          "containerPort": 2000,
          "protocol": "udp"
        }
      ]
    }
  ],
  "requiresCompatibilities" : [ "EC2" ],
  "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
  "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

進階主題

使用 App Mesh 的 Canary 部署

Canary 部署和版本可協助您在舊版本的應用程式與新部署的版本之間切換流量。它也會監控新部署版本的運作狀態。如果新版本有任何問題，Canary 部署可以自動將流量切換回舊版本。Canary 部署可讓您在應用程式版本之間切換流量，並擁有更多控制權。

如需如何使用 App Mesh 為 Amazon ECS 實作 Canary 部署的詳細資訊，請參閱使用 App Mesh 為 Amazon ECS 建立具有 Canary 部署的管道

> **Note**
>
> 如需 App Mesh 的更多範例和逐步解說，請參閱 App Mesh 範例儲存庫。

AWS App Mesh 和 Kubernetes 入門

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請造訪此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

當您使用適用於 Kubernetes 的 App Mesh 控制器 AWS App Mesh 與 Kubernetes 整合時，您可以管理 App Mesh 資源，例如網格、虛擬服務、虛擬節點、虛擬路由器，以及透過 Kubernetes 的路由。您也可以自動將 App Mesh 附屬容器映像新增至 Kubernetes Pod 規格。本教學課程會逐步引導您安裝適用於 Kubernetes 的 App Mesh 控制器，以啟用此整合。

控制器隨附下列 Kubernetes 自訂資源定義的部署：`meshes`、`virtual services`、`virtual nodes` 和 `virtual routers`。控制器會監看自訂資源的建立、修改和刪除，並透過 App Mesh API 變更對應的 App Mesh the section called “網格”、the section called “虛擬服務”、the section called “虛擬節點”the section called “虛擬閘道”the section called “閘道路由”、、 the section called “虛擬路由器” (包括 the section called “路由”) 資源。若要進一步了解或為控制器提供貢獻，請參閱 GitHub 專案。

控制器也會安裝 webhook，將下列容器注入到標示為您指定名稱的 Kubernetes Pod 中。

- App Mesh Envoy 代理 – Envoy 使用 App Mesh 控制平面中定義的組態來判斷傳送應用程式流量的位置。
- App Mesh 代理路由管理器 – 更新 Pod 網路命名空間中的`iptables`規則，以透過 Envoy 路由傳入和傳出流量。此容器會在 Pod 內以 Kubernetes 初始化容器執行。

先決條件

- 對 App Mesh 概念的現有理解。如需詳細資訊，請參閱什麼是 AWS App Mesh？。
- Kubernetes 概念的現有理解。如需詳細資訊，請參閱 Kubernetes 文件中的什麼是 Kubernetes。
- 現有 Kubernetes 叢集。如果您沒有現有的叢集，請參閱《Amazon EKS 使用者指南》中的 Amazon EKS 入門。如果您在 Amazon EC2 上執行自己的 Kubernetes 叢集，則請確保 Docker 已向 Envoy 映像所在的 Amazon ECR 儲存庫進行身分驗證。如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的 Envoy 映像、登錄檔身分驗證，以及《Kubernetes 文件》中的從私有登錄檔提取映像。

- App Mesh 支援已向 DNS 註冊的 Linux 服務 AWS Cloud Map，或兩者皆支援。若要使用此入門指南，建議您擁有三個已向 DNS 註冊的現有服務。本主題中的程序假設現有服務名為 `serviceA`、和 `serviceB`，`serviceBv2`並且所有服務都可以透過名為 的命名空間探索`apps.local`。

  即使服務不存在，您也可以建立服務網格及其資源，但在部署實際服務之前，您無法使用網格。
- AWS CLI 已安裝 1.18.116 版或更新版本，或 2.0.38 版或更新版本。若要安裝或升級 AWS CLI，請參閱安裝 AWS CLI。
- 已設定為與您的 Kubernetes 叢集通訊的 `kubectl` 用戶端。如果您使用的是 Amazon Elastic Kubernetes Service，您可以使用安裝`kubectl`和設定`kubeconfig`檔案的指示。
- 已安裝 Helm 3.0 版或更新版本。如果您尚未安裝 Helm，請參閱《Amazon EKS 使用者指南》中的將 Helm 與 Amazon EKS 搭配使用。
- Amazon EKS 目前僅支援 `IPv4_ONLY`和 `IPv6_ONLY` IP 偏好設定，因為 Amazon EKS 目前僅支援能夠僅提供`IPv4`流量或僅提供`IPv6`流量的 Pod。

其餘步驟假設實際服務名稱為 `serviceA`、`serviceB` 和 `serviceBv2`，而且所有服務皆可透過名稱為 `apps.local` 的命名空間探索。

步驟 1：安裝整合元件

將整合元件安裝到每個託管要與 App Mesh 搭配使用之 Pod 的叢集中一次。

安裝整合元件

1. 此程序的其餘步驟需有已安裝預先發行版本控制器的叢集。如果您已安裝發行前版本，或不確定您是否擁有 ，您可以下載並執行指令碼來檢查叢集上是否已安裝發行前版本。

   ```
   curl -o pre_upgrade_check.sh https://raw.githubusercontent.com/aws/eks-charts/
   master/stable/appmesh-controller/upgrade/pre_upgrade_check.sh
   sh ./pre_upgrade_check.sh
   ```

   如果指令碼傳回 `Your cluster is ready for upgrade. Please proceed to the installation instructions`，您可以繼續進行下個步驟。如果傳回不同的訊息，則您必須先完成升級步驟，才能繼續進行。如需升級發行前版本的詳細資訊，請參閱在 GitHub 上的 Upgrade (升級)。
2. 將 `eks-charts` 儲存庫新增到 Helm。

   ```
   helm repo add eks https://aws.github.io/eks-charts
   ```

3. 安裝 App Mesh Kubernetes 自訂資源定義 (CRD)。

```
kubectl apply -k "https://github.com/aws/eks-charts/stable/appmesh-controller/crds?ref=master"
```

4. 為控制器建立 Kubernetes 命名空間。

```
kubectl create ns appmesh-system
```

5. 設定下列變數，以便在稍後步驟中使用。以現有叢集的數值取代 *cluster-name* 和 *Region-code*。

```
export CLUSTER_NAME=cluster-name
export AWS_REGION=Region-code
```

6. (選用) 如果您要在 Fargate 上執行控制器，則必須建立 Fargate 設定檔。如果您尚未eksctl安裝 ，請參閱《Amazon EKS 使用者指南》中的安裝或升級eksctl。如果您想要使用主控台建立設定檔，請參閱《Amazon EKS 使用者指南》中的建立 Fargate 設定檔。

```
eksctl create fargateprofile --cluster $CLUSTER_NAME --name appmesh-system --namespace appmesh-system
```

7. 為您的叢集建立 OpenID Connect (OIDC) 身分識別提供者。如果您尚未eksctl安裝 ，您可以使用《Amazon EKS 使用者指南》中的安裝或升級eksctl說明進行安裝。如果您想要使用主控台建立提供者，請參閱《Amazon EKS 使用者指南》中的為叢集上的服務帳戶啟用 IAM 角色。

```
eksctl utils associate-iam-oidc-provider \
    --region=$AWS_REGION \
    --cluster $CLUSTER_NAME \
    --approve
```

8. 建立 IAM 角色，將 AWSAppMeshFullAccess和 AWSCloudMapFullAccess AWS 受管政策連接至該角色，並將其繫結至 Kubernetes appmesh-controller 服務帳戶。該角色可讓控制器新增、移除和變更 App Mesh 資源。

> **Note**
>
> 命令會建立具有自動產生名稱的 IAM AWS 角色。您無法指定所建立的 IAM 角色名稱。

```
eksctl create iamserviceaccount \
    --cluster $CLUSTER_NAME \
    --namespace appmesh-system \
    --name appmesh-controller \
    --attach-policy-arn  arn:aws:iam::aws:policy/
AWSCloudMapFullAccess,arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
    --override-existing-serviceaccounts \
    --approve
```

如果您想要使用 AWS Management Console 或 建立服務帳戶 AWS CLI，請參閱《Amazon EKS 使用者指南》中的為您的服務帳戶建立 IAM 角色和政策。如果您使用 AWS Management Console 或 AWS CLI 來建立帳戶，您也需要將角色映射至 Kubernetes 服務帳戶。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的為您的服務帳戶指定 IAM 角色。

9. 部署 App Mesh 控制器。如需所有組態選項的清單，請參閱 GitHub 上的組態 。

   1. 若要部署私有叢集的 App Mesh 控制器，您必須先將 App Mesh 和服務探索 Amazon VPC 端點啟用至連結的私有子網路。您也需要設定 `accountId`。

      ```
      --set accountId=$AWS_ACCOUNT_ID
      ```

      若要在私有叢集中啟用 X-Ray 追蹤，請啟用 X-Ray 和 Amazon ECR Amazon VPC 端點。根據`public.ecr.aws/xray/aws-xray-daemon:latest`預設，控制器會使用 ，因此請將此映像提取至本機，並將其推送至您的個人 ECR 儲存庫。

      > **Note**
      >
      > Amazon VPC 端點目前不支援 Amazon ECR 公有儲存庫。

      下列範例顯示使用 X-Ray 的組態部署控制器。

      ```
      helm upgrade -i appmesh-controller eks/appmesh-controller \
          --namespace appmesh-system \
          --set region=$AWS_REGION \
          --set serviceAccount.create=false \
          --set serviceAccount.name=appmesh-controller \
          --set accountId=$AWS_ACCOUNT_ID \
          --set log.level=debug \
      ```

```
    --set tracing.enabled=true \
    --set tracing.provider=x-ray \
    --set xray.image.repository=your-account-id.dkr.ecr.your-
region.amazonaws.com/your-repository \
    --set xray.image.tag=your-xray-daemon-image-tag
```

確認您的虛擬節點或閘道繫結應用程式部署時，是否成功插入 X-Ray 協助程式。

如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的私有叢集。

2. 部署其他叢集的 App Mesh 控制器。如需所有組態選項的清單，請參閱 GitHub 上的組態 。

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
    --namespace appmesh-system \
    --set region=$AWS_REGION \
    --set serviceAccount.create=false \
    --set serviceAccount.name=appmesh-controller
```

> **Note**
>
> 如果您的 Amazon EKS 叢集系列是 IPv6，請在部署 App Mesh 控制器時將下列選項新增至先前的命令 ，以設定叢集名稱`--set clusterName=$CLUSTER_NAME`。

> **Important**
>
> 如果您的叢集位於 me-south-1、ap-east-1、il-central-1、、 ap-southeast-3 eu-south-1或 af-south-1區域，則需要將下列選項新增至先前的命令：
> 將 *account-id* 和 *Region-code* 取代為一組適當的值。
>
> - 對於附屬影像：
>   - `--set image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/amazon/appmesh-controller`
>   - 772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod

    - 856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
    - 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
    - 422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
    - 564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
    - 924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
- 您可以在 GitHub 的變更日誌中找到較舊的影像 URIs。版本 中影像所在的 AWS 帳戶已變更`v1.5.0`。舊版映像託管在 AWS Amazon Elastic Kubernetes Service Amazon 容器映像登錄檔中找到的帳戶上。
- 對於控制器映像：
    - ```
      --set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
      ```
    - 772975370895.dkr.ecr.me-south-1.amazonaws.com/amazon/appmesh-controller:v1.13.1
    - 856666278305.dkr.ecr.ap-east-1.amazonaws.com/amazon/appmesh-controller:v1.13.1
    - 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/amazon/appmesh-controller:v1.13.1
    - 422531588944.dkr.ecr.eu-south-1.amazonaws.com/amazon/appmesh-controller:v1.13.1
    - 564877687649.dkr.ecr.il-central-1.amazonaws.com/amazon/appmesh-controller:v1.13.1
    - 924023996002.dkr.ecr.af-south-1.amazonaws.com/amazon/appmesh-controller:v1.13.1
- 對於附屬初始化映像：
    - ```
      --set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
      ```

- 772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-proxy-route-manager:v7-prod
- 856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-proxy-route-manager:v7-prod
- 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-proxy-route-manager:v7-prod
- 422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-proxy-route-manager:v7-prod
- 564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-proxy-route-manager:v7-prod
- 924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-proxy-route-manager:v7-prod

Important

僅支援 v1.9.0.0 版或更新版本搭配 App Mesh 使用。

10. 確認控制器版本為 `v1.4.0` 或更新版本。您可以在 GitHub 上查看 change log (變更日誌)。

```
kubectl get deployment appmesh-controller \
    -n appmesh-system \
    -o json  | jq -r ".spec.template.spec.containers[].image" | cut -f2 -d ':'
```

Note

如果您檢視執行中容器的日誌，您應該會看到一行文字，其中包含下列文字，而您可以放心忽略此內容。

```
Neither -kubeconfig nor -master was specified. Using the inClusterConfig.
 This might not work.
```

步驟 2：部署 App Mesh 資源

在 Kubernetes 中部署應用程式時，您也可以建立 Kubernetes 自訂資源，讓控制器可以建立對應的 App Mesh 資源。下列程序可協助您部署 App Mesh 資源及其部分功能。您可以在 GitHub 上 App Mesh 演練中列出的許多功能資料夾的`v1beta2`子資料夾中，找到部署其他 App Mesh 資源功能的範例資訊清單。 https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs

> **Important**
>
> 控制器建立 App Mesh 資源後，建議您只使用控制器變更或刪除 App Mesh 資源。如果您使用 App Mesh 變更或刪除資源，控制器預設不會變更或重新建立變更或刪除的 App Mesh 資源 10 小時。您可以將此持續時間設定為較短。如需詳細資訊，請參閱 GitHub 上的 Configuration (組態)。

部署 App Mesh 資源

1. 建立 Kubernetes 命名空間以部署 App Mesh 資源。

   a. 將下列內容儲存到電腦上名為 `namespace.yaml` 的檔案中。

   ```
   apiVersion: v1
   kind: Namespace
   metadata:
     name: my-apps
     labels:
       mesh: my-mesh
       appmesh.k8s.aws/sidecarInjectorWebhook: enabled
   ```

   b. 建立命名空間。

   ```
   kubectl apply -f namespace.yaml
   ```

2. 建立 App Mesh 服務網格。

   a. 將下列內容儲存到電腦上名為 `mesh.yaml` 的檔案中。檔案用於建立名為 的網格資源*my-mesh*。服務網格是在它之內各服務之間網路流量的邏輯邊界。

   ```
   apiVersion: appmesh.k8s.aws/v1beta2
   kind: Mesh
   metadata:
   ```

```
  name: my-mesh
spec:
  namespaceSelector:
    matchLabels:
      mesh: my-mesh
```

b. 建立網格。

```
kubectl apply -f mesh.yaml
```

c. 檢視已建立 Kubernetes 網格資源的詳細資料。

```
kubectl describe mesh my-mesh
```

輸出

```
Name:         my-mesh
Namespace:
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"Mesh","metadata":{"annotations":{},"name":"my-mesh"},"spec":
{"namespaceSelector":{"matchLa...
API Version:  appmesh.k8s.aws/v1beta2
Kind:         Mesh
Metadata:
  Creation Timestamp:  2020-06-17T14:51:37Z
  Finalizers:
    finalizers.appmesh.k8s.aws/mesh-members
    finalizers.appmesh.k8s.aws/aws-appmesh-resources
  Generation:        1
  Resource Version:  6295
  Self Link:         /apis/appmesh.k8s.aws/v1beta2/meshes/my-mesh
  UID:               111a11b1-c11d-1e1f-gh1i-j11k1l111m711
Spec:
  Aws Name:  my-mesh
  Namespace Selector:
    Match Labels:
      Mesh:  my-mesh
Status:
  Conditions:
    Last Transition Time:  2020-06-17T14:51:37Z
    Status:                True
```

```
    Type:                   MeshActive
  Mesh ARN:                 arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh
  Observed Generation:      1
Events:                     <none>
```

d. 檢視控制器建立之 App Mesh 服務網格的詳細資訊。

```
aws appmesh describe-mesh --mesh-name my-mesh
```

輸出

```
{
    "mesh": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh",
            "createdAt": "2020-06-17T09:51:37.920000-05:00",
            "lastUpdatedAt": "2020-06-17T09:51:37.920000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {},
        "status": {
            "status": "ACTIVE"
        }
    }
}
```

3. 建立 App Mesh 虛擬節點。虛擬節點會充當 Kubernetes 部署的邏輯指標。

a. 將下列內容儲存到電腦上名為 `virtual-node.yaml` 的檔案中。檔案用於在 *my-apps* 命名空間*my-service-a*中建立名為 的 App Mesh 虛擬節點。該虛擬節點代表在稍後的步驟中建立的 Kubernetes 服務。`hostname` 的值是此虛擬節點所代表之實際服務的完整 DNS 主機名稱。

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: my-service-a
  namespace: my-apps
```

```
spec:
  podSelector:
    matchLabels:
      app: my-app-1
  listeners:
    - portMapping:
        port: 80
        protocol: http
  serviceDiscovery:
    dns:
      hostname: my-service-a.my-apps.svc.cluster.local
```

虛擬節點具有本教學未涵蓋的功能，例如端對端加密和運作狀態檢查。如需詳細資訊，請參閱the section called “虛擬節點”。若要查看您可以為上述規格中的虛擬節點設定的所有可用設定，請執行下列命令。

```
aws appmesh create-virtual-node --generate-cli-skeleton yaml-input
```

b. 部署虛擬節點。

```
kubectl apply -f virtual-node.yaml
```

c. 檢視所建立之 Kubernetes 虛擬節點資源的詳細資料。

```
kubectl describe virtualnode my-service-a -n my-apps
```

輸出

```
Name:         my-service-a
Namespace:    my-apps
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualNode","metadata":{"annotations":{},"name":"my-service-
a","namespace":"my-apps"},"s...
API Version:  appmesh.k8s.aws/v1beta2
Kind:         VirtualNode
Metadata:
  Creation Timestamp:  2020-06-17T14:57:29Z
  Finalizers:
    finalizers.appmesh.k8s.aws/aws-appmesh-resources
  Generation:        2
```

```
  Resource Version:  22545
  Self Link:         /apis/appmesh.k8s.aws/v1beta2/namespaces/my-apps/
virtualnodes/my-service-a
  UID:               111a11b1-c11d-1e1f-gh1i-j11k1l111m711
Spec:
  Aws Name:  my-service-a_my-apps
  Listeners:
    Port Mapping:
      Port:      80
      Protocol:  http
  Mesh Ref:
    Name:  my-mesh
    UID:   111a11b1-c11d-1e1f-gh1i-j11k1l111m711
  Pod Selector:
    Match Labels:
      App:  nginx
  Service Discovery:
    Dns:
      Hostname:  my-service-a.my-apps.svc.cluster.local
Status:
  Conditions:
    Last Transition Time:  2020-06-17T14:57:29Z
    Status:                True
    Type:                  VirtualNodeActive
  Observed Generation:     2
  Virtual Node ARN:        arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps
Events:                    <none>
```

d. 檢視控制器在 App Mesh 中建立之虛擬節點的詳細資訊。

> **Note**
>
> 即使在 Kubernetes 中建立的虛擬節點名稱是 *my-service-a*，但在 App Mesh 中建立的虛擬節點名稱為 *my-service-a_my-apps*。控制器會在建立 App Mesh 資源時，將 Kubernetes 命名空間名稱附加至 App Mesh 虛擬節點名稱。新增命名空間名稱，因為在 Kubernetes 中，您可以在不同的命名空間中建立同名的虛擬節點，但在 App Mesh 中，虛擬節點名稱在網格中必須是唯一的。

```
aws appmesh describe-virtual-node --mesh-name my-mesh --virtual-node-name my-service-a_my-apps
```

輸出

```
{
    "virtualNode": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps",
            "createdAt": "2020-06-17T09:57:29.840000-05:00",
            "lastUpdatedAt": "2020-06-17T09:57:29.840000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {
            "backends": [],
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ],
            "serviceDiscovery": {
                "dns": {
                    "hostname": "my-service-a.my-apps.svc.cluster.local"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualNodeName": "my-service-a_my-apps"
    }
}
```

4. 建立 App Mesh 虛擬路由器。虛擬路由器會處理網格內一或多個虛擬服務的流量。

   a. 將下列內容儲存到電腦上名為 `virtual-router.yaml` 的檔案中。檔案用於建立虛擬路由器，將流量路由到在上一個步驟中建立的名為 `my-service-a` 的虛擬節點。控制器會建立 App Mesh 虛擬路由器和路由資源。您可以為路由指定更多功能，並使用除 `http` 以外的通訊協定。如需詳細資訊，請參閱the section called “虛擬路由器”及the section called “路由”。請注意，參考的虛擬節點名稱是 Kubernetes 虛擬節點名稱，而不是控制器在 App Mesh 中建立的 App Mesh 虛擬節點名稱。

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualRouter
metadata:
  namespace: my-apps
  name: my-service-a-virtual-router
spec:
  listeners:
    - portMapping:
        port: 80
        protocol: http
  routes:
    - name: my-service-a-route
      httpRoute:
        match:
          prefix: /
        action:
          weightedTargets:
            - virtualNodeRef:
                name: my-service-a
              weight: 1
```

   （選用）若要查看您可以在上述規格中設定的虛擬路由器的所有可用設定，請執行下列命令。

```
aws appmesh create-virtual-router --generate-cli-skeleton yaml-input
```

   若要查看您可以在上述規格中設定的路由的所有可用設定，請執行下列命令。

```
aws appmesh create-route --generate-cli-skeleton yaml-input
```

   b. 部署虛擬路由器。

```
kubectl apply -f virtual-router.yaml
```

c. 檢視已建立的 Kubernetes 的虛擬路由器資源。

```
kubectl describe virtualrouter my-service-a-virtual-router -n my-apps
```

縮寫輸出

```
Name:         my-service-a-virtual-router
Namespace:    my-apps
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualRouter","metadata":{"annotations":{},"name":"my-
service-a-virtual-router","namespac...
API Version:  appmesh.k8s.aws/v1beta2
Kind:         VirtualRouter
...
Spec:
  Aws Name:  my-service-a-virtual-router_my-apps
  Listeners:
    Port Mapping:
      Port:      80
      Protocol:  http
  Mesh Ref:
    Name:  my-mesh
    UID:   111a11b1-c11d-1e1f-gh1i-j11k1l111m711
  Routes:
    Http Route:
      Action:
        Weighted Targets:
          Virtual Node Ref:
            Name:  my-service-a
          Weight:  1
      Match:
        Prefix:  /
    Name:        my-service-a-route
Status:
  Conditions:
    Last Transition Time:  2020-06-17T15:14:01Z
    Status:                True
    Type:                  VirtualRouterActive
```

```
  Observed Generation:     1
  Route AR Ns:
    My - Service - A - Route:  arn:aws:appmesh:us-west-2:111122223333:mesh/my-
mesh/virtualRouter/my-service-a-virtual-router_my-apps/route/my-service-a-route
  Virtual Router ARN:          arn:aws:appmesh:us-west-2:111122223333:mesh/my-
mesh/virtualRouter/my-service-a-virtual-router_my-apps
Events:                        <none>
```

d. 檢視控制器在 App Mesh 中建立的虛擬路由器資源。您`my-service-a-virtual-router_my-apps`為 指定 name，因為當控制器在 App Mesh 中建立虛擬路由器時，會將 Kubernetes 命名空間名稱附加至虛擬路由器的名稱。

```
aws appmesh describe-virtual-router --virtual-router-name my-service-a-virtual-
router_my-apps --mesh-name my-mesh
```

輸出

```
{
    "virtualRouter": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualRouter/my-service-a-virtual-router_my-apps",
            "createdAt": "2020-06-17T10:14:01.547000-05:00",
            "lastUpdatedAt": "2020-06-17T10:14:01.547000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "ACTIVE"
```

```
        },
        "virtualRouterName": "my-service-a-virtual-router_my-apps"
    }
}
```

e. 檢視控制器在 App Mesh 中建立的路由資源。因為路由是 Kubernetes 中虛擬路由器組態的一部分，所以並未在 Kubernetes 中建立路由資源。路由資訊會顯示在 Kubernetes 資源詳情的子步驟 c 中。控制器在 App Mesh 中建立路由時，並未將 Kubernetes 命名空間名稱附加至 App Mesh 路由名稱，因為路由名稱對虛擬路由器是唯一的。

```
aws appmesh describe-route \
    --route-name my-service-a-route \
    --virtual-router-name my-service-a-virtual-router_my-apps \
    --mesh-name my-mesh
```

輸出

```
{
    "route": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualRouter/my-service-a-virtual-router_my-apps/route/my-service-a-route",
            "createdAt": "2020-06-17T10:14:01.577000-05:00",
            "lastUpdatedAt": "2020-06-17T10:14:01.577000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "routeName": "my-service-a-route",
        "spec": {
            "httpRoute": {
                "action": {
                    "weightedTargets": [
                        {
                            "virtualNode": "my-service-a_my-apps",
                            "weight": 1
                        }
                    ]
                },
                "match": {
```

```
                    "prefix": "/"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "my-service-a-virtual-router_my-apps"
    }
}
```

5. 建立 App Mesh 虛擬服務。虛擬服務是實際服務的抽象化，由虛擬節點直接提供，或透過虛擬路由器間接提供。相依服務會依其名稱呼叫您的虛擬服務。雖然名稱與 App Mesh 無關，但我們建議將虛擬服務命名為虛擬服務所代表之實際服務的完整網域名稱。透過這種方式命名您的虛擬服務，您不需要變更應用程式程式碼來參考不同的名稱。請求會路由傳送到指定為虛擬服務之提供者的虛擬節點或虛擬路由器。

   a. 將下列內容儲存到電腦上名為 `virtual-service.yaml` 的檔案中。檔案用於建立虛擬服務，該服務使用虛擬路由器供應商將流量路由到在上一個步驟中建立的名為 `my-service-a` 的虛擬節點。`spec` 中的 `awsName` 數值是此虛擬服務所抽象之實際 Kubernetes 服務的完整格式網域名稱 (FQDN)。在 the section called “步驟 3：建立或更新服務” 中建立 Kubernetes 服務。如需詳細資訊，請參閱the section called “虛擬服務”。

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualService
metadata:
  name: my-service-a
  namespace: my-apps
spec:
  awsName: my-service-a.my-apps.svc.cluster.local
  provider:
    virtualRouter:
      virtualRouterRef:
        name: my-service-a-virtual-router
```

   若要查看您可以為上述規格中的虛擬服務設定的所有可用設定，請執行下列命令。

```
aws appmesh create-virtual-service --generate-cli-skeleton yaml-input
```

   b. 建立虛擬服務。

```
kubectl apply -f virtual-service.yaml
```

c. 檢視所建立之 Kubernetes 虛擬服務資源的詳細資料。

```
kubectl describe virtualservice my-service-a -n my-apps
```

輸出

```
Name:         my-service-a
Namespace:    my-apps
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualService","metadata":{"annotations":{},"name":"my-
service-a","namespace":"my-apps"}...
API Version:  appmesh.k8s.aws/v1beta2
Kind:         VirtualService
Metadata:
  Creation Timestamp:  2020-06-17T15:48:40Z
  Finalizers:
    finalizers.appmesh.k8s.aws/aws-appmesh-resources
  Generation:        1
  Resource Version:  13598
  Self Link:         /apis/appmesh.k8s.aws/v1beta2/namespaces/my-apps/
virtualservices/my-service-a
  UID:               111a11b1-c11d-1e1f-gh1i-j11k1l111m711
Spec:
  Aws Name:  my-service-a.my-apps.svc.cluster.local
  Mesh Ref:
    Name:  my-mesh
    UID:   111a11b1-c11d-1e1f-gh1i-j11k1l111m711
  Provider:
    Virtual Router:
      Virtual Router Ref:
        Name:  my-service-a-virtual-router
Status:
  Conditions:
    Last Transition Time:  2020-06-17T15:48:40Z
    Status:                True
    Type:                  VirtualServiceActive
  Observed Generation:     1
```

```
  Virtual Service ARN:     arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualService/my-service-a.my-apps.svc.cluster.local
Events:                  <none>
```

d. 檢視控制器在 App Mesh 中建立的虛擬服務資源詳細資訊。Kubernetes 控制器在 App Mesh 中建立虛擬服務時，並未將 Kubernetes 命名空間名稱附加至 App Mesh 虛擬服務名稱，因為虛擬服務的名稱是唯一的 FQDN。

```
aws appmesh describe-virtual-service --virtual-service-name my-service-a.my-
apps.svc.cluster.local --mesh-name my-mesh
```

輸出

```
{
    "virtualService": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualService/my-service-a.my-apps.svc.cluster.local",
            "createdAt": "2020-06-17T10:48:40.182000-05:00",
            "lastUpdatedAt": "2020-06-17T10:48:40.182000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {
            "provider": {
                "virtualRouter": {
                    "virtualRouterName": "my-service-a-virtual-router_my-apps"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualServiceName": "my-service-a.my-apps.svc.cluster.local"
    }
}
```

雖然本教學未涵蓋，但控制器也可以部署 App Mesh the section called “虛擬閘道”和 the section called “閘道路由”。如需使用控制器部署這些資源的逐步解說，請參閱設定傳入閘道，或包含 GitHub 資源的範例資訊清單。

步驟 3：建立或更新服務

您想要與 App Mesh 搭配使用的任何 Pod 都必須新增 App Mesh 附屬容器。注入器會自動將附屬容器新增至部署您指定命名空間的任何 Pod 中。

1. 啟用代理伺服器授權。我們建議您啟用每個 Kubernetes 部署，以僅串流其 App Mesh 虛擬節點的組態。

   a. 將下列內容儲存到電腦上名為 `proxy-auth.json` 的檔案中。請務必用您自己的*#####*取代。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "appmesh:StreamAggregatedResources",
               "Resource": [
                   "arn:aws:appmesh:Region-code:111122223333:mesh/my-mesh/
   virtualNode/my-service-a_my-apps"
               ]
           }
       ]
   }
   ```

   b. 建立政策。

   ```
   aws iam create-policy --policy-name my-policy --policy-document file://proxy-
   auth.json
   ```

   c. 建立 IAM 角色、將您在上一個步驟中建立的政策連接至該角色、建立 Kubernetes 服務帳戶，並將政策繫結至 Kubernetes 服務帳戶。該角色可讓控制器新增、移除和變更 App Mesh 資源。

   ```
   eksctl create iamserviceaccount \
       --cluster $CLUSTER_NAME \
       --namespace my-apps \
       --name my-service-a \
   ```

```
    --attach-policy-arn  arn:aws:iam::111122223333:policy/my-policy \
    --override-existing-serviceaccounts \
    --approve
```

如果您想要使用 AWS Management Console 或 建立服務帳戶 AWS CLI，請參閱《Amazon EKS 使用者指南》中的為您的服務帳戶建立 IAM 角色和政策。如果您使用 AWS Management Console 或 AWS CLI 來建立帳戶，您也需要將角色映射至 Kubernetes 服務帳戶。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的為您的服務帳戶指定 IAM 角色。

2. (選用) 如果您要將您的部署內容部署到 Fargate Pod，則需要建立 Fargate 設定檔。如果您尚未eksctl安裝 ，您可以使用 Amazon EKS 使用者指南中的安裝或升級eksctl中的指示進行安裝。如果您想要使用主控台建立設定檔，請參閱《Amazon EKS 使用者指南》中的建立 Fargate 設定檔。

```
eksctl create fargateprofile --cluster my-cluster --region Region-code --name my-service-a --namespace my-apps
```

3. 建立 Kubernetes 服務和部署。如果您有要與 App Mesh 搭配使用的現有部署，則需要部署虛擬節點，就像在 3的子步驟中一樣the section called “步驟 2：部署 App Mesh 資源”。更新您的部署，以確保其標籤符合您在虛擬節點上設定的標籤，以便附屬容器自動新增至 Pod，並重新部署 Pod。

   a. 將下列內容儲存到電腦上名為 example-service.yaml 的檔案中。如果您變更命名空間名稱並使用 Fargate Pod，請確定命名空間名稱符合您在 Fargate 設定檔中定義的命名空間名稱。

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-a
  namespace: my-apps
  labels:
    app: my-app-1
spec:
  selector:
    app: my-app-1
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-service-a
  namespace: my-apps
  labels:
    app: my-app-1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app-1
  template:
    metadata:
      labels:
        app: my-app-1
    spec:
      serviceAccountName: my-service-a
      containers:
      - name: nginx
        image: nginx:1.19.0
        ports:
        - containerPort: 80
```

> **Important**
>
> 規範中的 app matchLabels selector 值必須與您在 the section called “步驟 2：部署 App Mesh 資源” 的子步驟 3 中建立虛擬節點時指定的值相符，否則附屬容器將不會注入到 Pod 中。在上一個範例中，標籤的值為 my-app-1。如果您部署虛擬閘道，而不是虛擬節點，則Deployment資訊清單應該只包含 Envoy 容器。如需要使用之映像的詳細資訊，請參閱 Envoy。如需範例手冊節，請參閱 GitHub 上的部署範例。

b. 部署服務。

```
kubectl apply -f example-service.yaml
```

c. 檢視服務和部署。

```
kubectl -n my-apps get pods
```

輸出

```
NAME                              READY   STATUS    RESTARTS   AGE
my-service-a-54776556f6-2cxd9     2/2     Running   0          10s
my-service-a-54776556f6-w26kf     2/2     Running   0          18s
my-service-a-54776556f6-zw5kt     2/2     Running   0          26s
```

d. 檢視已部署之其中一個 pod 的詳細資訊。

```
kubectl -n my-apps describe pod my-service-a-54776556f6-2cxd9
```

縮寫輸出

```
Name:         my-service-a-54776556f6-2cxd9
Namespace:    my-app-1
Priority:     0
Node:         ip-192-168-44-157.us-west-2.compute.internal/192.168.44.157
Start Time:   Wed, 17 Jun 2020 11:08:59 -0500
Labels:       app=nginx
              pod-template-hash=54776556f6
Annotations:  kubernetes.io/psp: eks.privileged
Status:       Running
IP:           192.168.57.134
IPs:
  IP:           192.168.57.134
Controlled By:  ReplicaSet/my-service-a-54776556f6
Init Containers:
  proxyinit:
    Container ID:   docker://
e0c4810d584c21ae0cb6e40f6119d2508f029094d0e01c9411c6cf2a32d77a59
    Image:          111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
proxy-route-manager:v2
    Image ID:       docker-pullable://111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager
    Port:           <none>
    Host Port:      <none>
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Fri, 26 Jun 2020 08:36:22 -0500
      Finished:     Fri, 26 Jun 2020 08:36:22 -0500
    Ready:          True
```

```
    Restart Count:  0
    Requests:
      cpu:     10m
      memory:  32Mi
    Environment:
      APPMESH_START_ENABLED:         1
      APPMESH_IGNORE_UID:            1337
      APPMESH_ENVOY_INGRESS_PORT:    15000
      APPMESH_ENVOY_EGRESS_PORT:     15001
      APPMESH_APP_PORTS:             80
      APPMESH_EGRESS_IGNORED_IP:     169.254.169.254
      APPMESH_EGRESS_IGNORED_PORTS:  22
      AWS_ROLE_ARN:                  arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
      AWS_WEB_IDENTITY_TOKEN_FILE:   /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
    ...
Containers:
  nginx:
    Container ID:   docker://
be6359dc6ecd3f18a1c87df7b57c2093e1f9db17d5b3a77f22585ce3bcab137a
    Image:          nginx:1.19.0
    Image ID:       docker-pullable://nginx
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 26 Jun 2020 08:36:28 -0500
    Ready:          True
    Restart Count:  0
    Environment:
      AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
    ...
  envoy:
    Container ID:
 docker://905b55cbf33ef3b3debc51cb448401d24e2e7c2dbfc6a9754a2c49dd55a216b6
    Image:          840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.12.4.0-prod
    Image ID:       docker-pullable://840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy
    Port:           9901/TCP
    Host Port:      0/TCP
```

```
    State:          Running
      Started:      Fri, 26 Jun 2020 08:36:36 -0500
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:     10m
      memory:  32Mi
    Environment:
      APPMESH_RESOURCE_ARN:         arn:aws:iam::111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps
      APPMESH_PREVIEW:              0
      ENVOY_LOG_LEVEL:              info
      AWS_REGION:                   us-west-2
      AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
...
Events:
  Type    Reason     Age   From
   Message
  ----    ------     ----  ----
   -------
  Normal  Pulling    30s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Pulling image "111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager:v2"
  Normal  Pulled     23s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Successfully pulled image "111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager:v2"
  Normal  Created    21s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Created container proxyinit
  Normal  Started    21s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Started container proxyinit
  Normal  Pulling    20s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Pulling image "nginx:1.19.0"
  Normal  Pulled     16s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Successfully pulled image "nginx:1.19.0"
  Normal  Created    15s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Created container nginx
  Normal  Started    15s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Started container nginx
  Normal  Pulling    15s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Pulling image "840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy:v1.12.4.0-prod"
```

```
  Normal  Pulled     8s    kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Successfully pulled image "840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy:v1.12.4.0-prod"
  Normal  Created    7s    kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Created container envoy
  Normal  Started    7s    kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Started container envoy
```

在前面的輸出中，您可以看到 proxyinit 和 envoy 容器已新增至 pod 中。如果您已將範例服務部署至 Fargate，則控制器會將envoy容器新增至 Pod，但容器proxyinit未新增。

4. （選用）安裝附加元件，例如 Prometheus、Grafana AWS X-Ray、Jaeger 和 Datadog。如需詳細資訊，請參閱 GitHub 上的 App Mesh 附加元件和 App Mesh 使用者指南中的可觀測性一節。

> **Note**
>
> 如需 App Mesh 的更多範例和逐步解說，請參閱 App Mesh 範例儲存庫。

步驟 4：清理

移除本教學中建立的所有範例資源。控制器也會移除在 my-mesh App Mesh 服務網格中建立的資源。

```
kubectl delete namespace my-apps
```

如果您已為範例服務建立 Fargate 設定檔，請將其移除。

```
eksctl delete fargateprofile --name my-service-a --cluster my-cluster --region Region-
code
```

刪除網格。

```
kubectl delete mesh my-mesh
```

(選擇性) 您可以移除 Kubernetes 整合元件。

```
helm delete appmesh-controller -n appmesh-system
```

（選用）如果您將 Kubernetes 整合元件部署至 Fargate，請刪除 Fargate 設定檔。

```
eksctl delete fargateprofile --name appmesh-system --cluster my-cluster --
region Region-code
```

AWS App Mesh 和 Amazon EC2 入門

> **Important**
>
> 終止支援通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題可協助您 AWS App Mesh 在 Amazon EC2 上執行的實際服務使用 。本教學課程涵蓋數種 App Mesh 資源類型的基本功能。

案例

若要說明如何使用 App Mesh，假設您擁有具有下列特性的應用程式：

- 包含兩個名為 `serviceA`和 的服務`serviceB`。
- 這兩個服務都註冊到名稱為 `apps.local` 的命名空間。
- `ServiceA` 與 `serviceB` 透過 HTTP/2，連接埠 80 進行通訊。
- 您已部署 `serviceB` 第 2 版，並在 `apps.local` 命名空間中將其註冊為名稱 `serviceBv2`。

您有以下要求：

- 您想要將 75% 的流量從 `serviceA` 傳送到 `serviceB`，並將 25% 的流量傳送到 `serviceBv2`。透過僅將 25% 傳送到 `serviceBv2`，您可以在從 傳送 100% 的流量之前，驗證它是否沒有錯誤`serviceA`。
- 您希望能夠輕鬆地調整流量權重，以便證實流量可靠之後，能夠 100% 流入 `serviceBv2`。將所有流量傳送到 後`serviceBv2`，您想要停止 `serviceB`。
- 您不想變更實際服務的任何現有應用程式碼或服務探索註冊，以符合先前的要求。

為了滿足您的需求，您決定建立 App Mesh 服務網格，其中包含虛擬服務、虛擬節點、虛擬路由器和路由。實作網格後，您會更新您的服務以使用 Envoy 代理。一旦更新，您的服務會透過 Envoy 代理彼此通訊，而非直接相互通訊。

先決條件

App Mesh 支援已向 DNS 註冊的 Linux 服務 AWS Cloud Map，或兩者皆支援。若要使用此入門指南，建議您擁有三個已向 DNS 註冊的現有服務。即使服務不存在，您也可以建立服務網格及其資源，但在部署實際服務之前，您無法使用網格。

如果您還沒有執行中的服務，您可以啟動 Amazon EC2 執行個體並將應用程式部署到這些執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的教學課程：Amazon EC2 Linux 執行個體入門。 Amazon EC2 其餘步驟假設實際服務名稱為 `serviceA`、`serviceB` 和 `serviceBv2`，而且所有服務皆可透過名稱為 `apps.local` 的命名空間探索。

步驟 1：建立網格和虛擬服務

服務網格是在它之內各服務之間網路流量的邏輯邊界。如需詳細資訊，請參閱服務網格。虛擬服務是實際服務的抽象化。如需詳細資訊，請參閱虛擬服務。

建立下列資源：

- 名稱為 `apps` 的網格，因為案例中的所有服務皆註冊到 `apps.local` 命名空間。
- 名稱為 `serviceb.apps.local` 的虛擬服務，因為虛擬服務代表可使用該名稱探索的服務，而且您不想將程式碼變更為參照其他名稱。稍後的步驟會新增名稱為 `servicea.apps.local` 的虛擬服務。

您可以使用 AWS Management Console 或 1.18.116 AWS CLI 版或更新版本，或 2.0.38 版或更新版本來完成下列步驟。如果使用 AWS CLI，請使用 `aws --version`命令來檢查已安裝的 AWS CLI 版本。如果您沒有安裝 1.18.116 版或更新版本或 2.0.38 版或更新版本，則必須安裝或更新 AWS CLI。為您要使用的工具選取索引標籤。

AWS Management Console

1. 開啟 App Mesh 主控台的初次執行精靈，網址為 https：//https://console.aws.amazon.com/appmesh/get-started。
2. 對於 Mesh name (網格名稱)，輸入 **apps**。
3. 對於 Virtual service name (虛擬服務名稱)，輸入 **serviceb.apps.local**。
4. 若要繼續，請選擇 Next (下一步)。

AWS CLI

1. 使用 create-mesh 命令建立網格。

```
aws appmesh create-mesh --mesh-name apps
```

2. 使用 create-virtual-service 命令建立虛擬服務。

```
aws appmesh create-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local --spec {}
```

步驟 2：建立虛擬節點

虛擬節點可做為實際服務的邏輯指標。如需詳細資訊，請參閱虛擬節點。

建立名稱為 serviceB 的虛擬節點，因為其中一個虛擬節點代表名稱為 serviceB 的實際服務。虛擬節點代表的實際服務可透過 DNS (主機名稱為 serviceb.apps.local) 探索。或者，您可以使用 AWS Cloud Map探索實際服務。虛擬節點會使用連接埠 80 上的 HTTP/2 通訊協定接聽流量。也支援其他通訊協定，以及運作狀態檢查。您可以在後續步驟serviceBv2中為 serviceA和 建立虛擬節點。

AWS Management Console

1. 對於 Virtual node name (虛擬節點名稱)，輸入 **serviceB**。
2. 針對服務探索方法，選擇 DNS，然後輸入 **serviceb.apps.local** DNS 主機名稱。
3. 在接聽程式組態下，選擇通訊協定的 http2，然後輸入連接埠**80**的 。
4. 若要繼續，請選擇 Next (下一步)。

AWS CLI

1. 使用下列內容建立名為 create-virtual-node-serviceb.json 的檔案：

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
```

```
                    "protocol": "http2"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceB.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceB"
}
```

2. 使用 JSON 檔案做為輸入，搭配 create-virtual-node 命令建立虛擬節點。

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-serviceb.json
```

步驟 3：建立虛擬路由器和路由

虛擬路由器會路由網格內一或多個虛擬服務的流量。如需詳細資訊，請參閱虛擬路由器及路由。

建立下列資源：

- 名為 `serviceB` 的虛擬路由器，因為 `serviceB.apps.local` 虛擬服務不會啟動與任何其他服務的對外通訊。請記住，您先前建立的虛擬服務是實際 `serviceb.apps.local` 服務的抽象。虛擬服務會將流量傳送至虛擬路由器。虛擬路由器會使用連接埠 80 上的 HTTP/2 通訊協定接聽流量。也支援其他通訊協定。
- 名為 `serviceB` 的路由。它將 100% 的流量路由到`serviceB`虛擬節點。新增`serviceBv2`虛擬節點後，權重會進入後續步驟。雖然未涵蓋在本指南中，但您可以為路由新增其他篩選條件，並新增重試政策，使 Envoy 代理在遇到通訊問題時多次嘗試將流量傳送至虛擬節點。

AWS Management Console

1. 對於 Virtual router name (虛擬路由器名稱)，輸入 **`serviceB`**。
2. 在接聽程式組態下，為通訊協定選擇 http2，並為連接埠指定 **`80`** 。
3. 對於 Route name (路由名稱)，輸入 **`serviceB`**。
4. 對於 Route type (路由類型)，選擇 http2。

5. 針對目標組態下的虛擬節點名稱，選取 serviceB並輸入 **100**表示權重。
6. 在相符組態下，選擇方法。
7. 若要繼續，請選擇 Next (下一步)。

AWS CLI

1. 建立虛擬路由器。

   a. 使用下列內容建立名為 create-virtual-router.json 的檔案：

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ]
    },
    "virtualRouterName": "serviceB"
}
```

   b. 使用 JSON 檔案做為輸入，搭配 create-virtual-router 命令建立虛擬路由器。

```
aws appmesh create-virtual-router --cli-input-json file://create-virtual-
router.json
```

2. 建立路由。

   a. 使用下列內容建立名為 create-route.json 的檔案：

```
{
    "meshName" : "apps",
    "routeName" : "serviceB",
    "spec" : {
        "httpRoute" : {
            "action" : {
                "weightedTargets" : [
```

```
                    {
                        "virtualNode" : "serviceB",
                        "weight" : 100
                    }
                ]
            },
            "match" : {
                "prefix" : "/"
            }
        }
    },
    "virtualRouterName" : "serviceB"
}
```

b. 使用 JSON 做為輸入，搭配 create-route 命令建立路由。

```
aws appmesh create-route --cli-input-json file://create-route.json
```

步驟 4：檢閱和建立

依照先前的指示檢閱設定。

AWS Management Console

如果您需要在任何區段中進行變更，請選擇 Edit (編輯)。對這些設定感到滿意後，請選擇 Create mesh (建立網格)。

Status (狀態) 畫面會顯示所有已建立的網格資源。選取 View mesh (檢視網格) 即可在主控台中檢視建立的資源。

AWS CLI

檢閱您使用 describe-mesh 命令建立的網格設定。

```
aws appmesh describe-mesh --mesh-name apps
```

檢閱您使用 describe-virtual-service 命令建立的虛擬服務設定。

```
aws appmesh describe-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local
```

檢閱您使用 describe-virtual-node 命令建立的虛擬節點設定。

```
aws appmesh describe-virtual-node --mesh-name apps --virtual-node-name serviceB
```

檢閱您使用 describe-virtual-router 命令建立的虛擬路由器設定。

```
aws appmesh describe-virtual-router --mesh-name apps --virtual-router-name serviceB
```

檢閱您使用 describe-route 命令建立的路由設定。

```
aws appmesh describe-route --mesh-name apps \
    --virtual-router-name serviceB  --route-name serviceB
```

步驟 5：建立其他資源

若要完成案例，您必須：

- 建立一個名為 serviceBv2 的虛擬節點，以及另一個名為 serviceA 的虛擬節點。這兩個虛擬節點都會透過 HTTP/2 連接埠 80 接聽請求。針對serviceA虛擬節點，設定 的後端serviceb.apps.local。來自serviceA虛擬節點的所有傳出流量都會傳送至名為 的虛擬服務serviceb.apps.local。雖然未涵蓋在本指南中，但您也可以指定將虛擬節點的存取日誌寫入其中的檔案路徑。
- 建立名為 的額外虛擬服務servicea.apps.local，將所有流量直接傳送至serviceA虛擬節點。
- 更新您在上一個步驟中建立的 serviceB 路由，將 75% 的流量傳送到 serviceB 虛擬節點，以及 25% 的流量傳送到 serviceBv2 虛擬節點。隨著時間的推移，您可以繼續修改權重，直到 serviceBv2 收到 100% 的流量為止。將所有流量傳送到 後serviceBv2，您可以關閉並停止serviceB虛擬節點和實際服務。當您更改權重時，您的程式碼不需要任何修改，因為 serviceb.apps.local 虛擬和實際服務名稱不會變更。記住，serviceb.apps.local 虛擬服務會將流量傳送至虛擬路由器，接著虛擬路由器會將流量路由至虛擬節點。虛擬節點的服務探索名稱可以隨時變更。

AWS Management Console

1. 在左側導覽窗格中，選取 Meshes (網格)。
2. 選取您在上一個步驟中建立的 apps 網格。
3. 在左側導覽窗格中，選取 Virtual nodes (虛擬節點)。
4. 選擇 Create virtual node (建立虛擬節點)。

5. 對於 Virtual node name (虛擬節點名稱) 輸入 **serviceBv2**、對於 Service discovery method (服務探索方法) 選擇 DNS，然後對於 DNS hostname (DNS 主機名稱) 輸入 **servicebv2.apps.local**。
6. 針對接聽程式組態，針對通訊協定選取 http2，然後**80**針對連接埠輸入 。
7. 選擇 Create virtual node (建立虛擬節點)。
8. 再次選擇 Create virtual node (建立虛擬節點)。輸入 **serviceA** 做為虛擬節點名稱。對於 Service discovery method (服務探索方法)，選擇 DNS，並針對 DNS hostname (DNS 主機名稱) 輸入 **servicea.apps.local**。
9. 在新增後端下輸入虛擬服務名稱，輸入 **serviceb.apps.local**。
10. 在接聽程式組態下，選擇通訊協定的 http2，在連接埠**80**中輸入 ，然後選擇建立虛擬節點。
11. 在左側導覽窗格中，選取 Virtual routers (虛擬路由器)，然後從清單中選取 serviceB 虛擬路由器。
12. 在 Routes (路由) 下方，選取您在上一個步驟中建立的路由 (名為 ServiceB)，然後選擇 Edit (編輯)。
13. 在目標、虛擬節點名稱下，將 的權重值變更為 serviceB **75**。
14. 選擇新增目標，serviceBv2從下拉式清單中選擇，並將權重的值設定為 **25**。
15. 選擇儲存。
16. 在左側瀏覽窗格中，選取 Virtual services (虛擬服務)，然後選擇 Create virtual service (建立虛擬服務)。
17. **servicea.apps.local** 針對虛擬服務名稱輸入 ，針對提供者選取虛擬節點，serviceA針對虛擬節點選取 ，然後選擇建立虛擬服務。

AWS CLI

1. 建立 serviceBv2 虛擬節點。

   a. 使用下列內容建立名為 create-virtual-node-servicebv2.json 的檔案：

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
```

```
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceBv2.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceBv2"
}
```

b. 建立虛擬節點。

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicebv2.json
```

2. 建立 serviceA 虛擬節點。

a. 使用下列內容建立名為 create-virtual-node-servicea.json 的檔案：

```
{
   "meshName" : "apps",
   "spec" : {
      "backends" : [
         {
            "virtualService" : {
               "virtualServiceName" : "serviceb.apps.local"
            }
         }
      ],
      "listeners" : [
         {
            "portMapping" : {
               "port" : 80,
               "protocol" : "http2"
            }
         }
      ],
      "serviceDiscovery" : {
         "dns" : {
            "hostname" : "servicea.apps.local"
         }
```

```
      }
   },
   "virtualNodeName" : "serviceA"
}
```

b. 建立虛擬節點。

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicea.json
```

3. 更新您在上一個步驟中建立的 serviceb.apps.local 虛擬服務，將其流量傳送至 serviceB 虛擬路由器。最初建立虛擬服務時，它不會傳送流量到任何地方，因為 serviceB 虛擬路由器尚未建立。

a. 使用下列內容建立名為 update-virtual-service.json 的檔案：

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualRouter" : {
            "virtualRouterName" : "serviceB"
         }
      }
   },
   "virtualServiceName" : "serviceb.apps.local"
}
```

b. 使用 update-virtual-service 命令更新虛擬服務。

```
aws appmesh update-virtual-service --cli-input-json file://update-virtual-service.json
```

4. 更新您在上一個步驟中建立的 serviceB 路由。

a. 使用下列內容建立名為 update-route.json 的檔案：

```
{
   "meshName" : "apps",
   "routeName" : "serviceB",
   "spec" : {
      "http2Route" : {
         "action" : {
```

```
            "weightedTargets" : [
               {
                  "virtualNode" : "serviceB",
                  "weight" : 75
               },
               {
                  "virtualNode" : "serviceBv2",
                  "weight" : 25
               }
            ]
         },
         "match" : {
            "prefix" : "/"
         }
      }
   },
   "virtualRouterName" : "serviceB"
}
```

b. 使用 update-route 命令更新路由。

```
aws appmesh update-route --cli-input-json file://update-route.json
```

5. 建立 serviceA 虛擬服務。

a. 使用下列內容建立名為 create-virtual-servicea.json 的檔案：

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualNode" : {
            "virtualNodeName" : "serviceA"
         }
      }
   },
   "virtualServiceName" : "servicea.apps.local"
}
```

b. 建立虛擬服務。

```
aws appmesh create-virtual-service --cli-input-json file://create-virtual-
servicea.json
```

網格摘要

在您建立服務網格之前，您有三個名稱為 servicea.apps.local、serviceb.apps.local 和 servicebv2.apps.local 的實際服務。除了實際服務以外，您目前擁有包含代表實際服務之下列資源的服務網格：

- 兩個虛擬服務。代理會透過虛擬路由器將 servicea.apps.local 虛擬服務的所有流量傳送至 serviceb.apps.local 虛擬服務。
- 名稱為 serviceA、serviceB 和 serviceBv2 的三個虛擬節點。Envoy 代理會使用針對虛擬節點設定的服務探索資訊，來查詢實際服務的 IP 地址。
- 具有單一路由的虛擬路由器，其指示 Envoy 代理將 75% 的傳入流量路由到 serviceB 虛擬節點，將 25% 的流量路由到 serviceBv2 虛擬節點。

步驟 6：更新服務

在建立網格之後，您需要完成下列任務：

- 授權您隨每個服務一起部署的 Envoy 代理，以讀取一或多個虛擬節點的組態。如需如何授權代理的詳細資訊，請參閱 Envoy Proxy 授權。
- 若要更新現有的服務，請完成以下步驟。

將 Amazon EC2 執行個體設定為虛擬節點成員

1. 建立 IAM 角色。

   a. 使用下列內容建立名為 ec2-trust-relationship.json 的檔案。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. 使用下列命令建立 IAM 角色。

```
aws iam create-role --role-name mesh-virtual-node-service-b --assume-role-
policy-document file://ec2-trust-relationship.json
```

2. 將 IAM 政策連接至角色，允許其從 Amazon ECR 讀取，並只讀取特定 App Mesh 虛擬節點的組態。

a. 使用下列內容建立名為 `virtual-node-policy.json` 的檔案。apps 是您在 the section called “步驟 1：建立網格和虛擬服務” 建立的網格名稱，而 `serviceB` 是您在 the section called “步驟 2：建立虛擬節點” 建立的虛擬節點名稱。將 *111122223333* 取代為您的帳戶 ID，將 *us-west-2* 取代為您建立網格的區域。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:StreamAggregatedResources",
            "Resource": [
                "arn:aws:appmesh:us-west-2:111122223333:mesh/apps/
virtualNode/serviceB"
            ]
        }
    ]
}
```

b. 使用下列命令建立政策。

```
aws iam create-policy --policy-name virtual-node-policy --policy-document
 file://virtual-node-policy.json
```

c. 將您在上一個步驟中建立的政策連接至角色，讓角色只能從 App Mesh 讀取`serviceB`虛擬節點的組態。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::111122223333:policy/
virtual-node-policy --role-name mesh-virtual-node-service-b
```

d. 將 `AmazonEC2ContainerRegistryReadOnly`受管政策連接至角色，以便從 Amazon ECR 提取 Envoy 容器映像。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name mesh-virtual-node-service-b
```

3. 使用您建立的 IAM 角色啟動 Amazon EC2 執行個體。
4. 透過 SSH 連線至您的執行個體。
5. 根據您的作業系統文件，在您的執行個體 AWS CLI 上安裝 Docker 和 。
6. 在您希望 Docker 用戶端從中提取映像的區域中，向 Envoy Amazon ECR 儲存庫進行身分驗證。

   - 除了 me-south-1、ap-east-1、ap-southeast-3、il-central-1、 eu-south-1和以外的所有區域af-south-1。您可以將 *us-west-2* 取代為任何支援的 區域，但 me-south-1、ap-east-1、ap-southeast-3eu-south-1、、 il-central-1和 除外af-south-1。

     ```
     $aws ecr get-login-password \
         --region us-west-2 \
     | docker login \
         --username AWS \
         --password-stdin 840364872350.dkr.ecr.us-west-2.amazonaws.com
     ```

   - me-south-1 區域

     ```
     $aws ecr get-login-password \
         --region me-south-1 \
     | docker login \
         --username AWS \
         --password-stdin 772975370895.dkr.ecr.me-south-1.amazonaws.com
     ```

   - ap-east-1 區域

     ```
     $aws ecr get-login-password \
         --region ap-east-1 \
     | docker login \
         --username AWS \
         --password-stdin 856666278305.dkr.ecr.ap-east-1.amazonaws.com
     ```

7. 執行下列其中一個命令，以啟動執行個體上的 App Mesh Envoy 容器，視您要從哪個區域提取映像而定。*apps* 和 *serviceB* 值是案例中定義的網格和虛擬節點名稱。此資訊會告知代理程式要從 App Mesh 讀取哪些虛擬節點組態。若要完成案例，您也需要為託管

serviceBv2和serviceA虛擬節點所代表服務的 Amazon EC2 執行個體完成這些步驟。對於您自己的應用程式，將這些值取代為您自己的值。

- 除了 me-south-1、ap-east-1、ap-southeast-3、il-central-1、 eu-south-1和以外的所有區域af-south-1。您可以將*###*取代為 me-south-1、ap-east-1、ap-southeast-3、il-central-1、 eu-south-1和 區域以外的任何支援af-south-1區域。您可以將 *1337* 換成 0 到 2147483647 之間的任何值。

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB  \
-u 1337 --network host 840364872350.dkr.ecr.region-code.amazonaws.com/aws-
appmesh-envoy:v1.29.12.1-prod
```

- me-south-1 區域。您可以將 *1337* 換成 0 到 2147483647 之間的任何值。

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB  \
-u 1337 --network host 772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-
envoy:v1.29.12.1-prod
```

- ap-east-1 區域。您可以將 *1337* 換成 0 到 2147483647 之間的任何值。

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB  \
-u 1337 --network host 856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-
envoy:v1.29.12.1-prod
```

> **Note**
>
> APPMESH_RESOURCE_ARN 屬性需要版本 1.15.0或更新版本的 Envoy 映像。如需詳細資訊，請參閱Envoy。

> **Important**
>
> 僅支援 v1.9.0.0 版或更新版本搭配 App Mesh 使用。

8. 請在下方選取 Show more。使用下列命令在您的執行個體上建立名為 envoy-networking.sh 的檔案。將 *8000* 取代為您的應用程式程式碼用於傳入流量的連接埠。您可以變更

APPMESH_IGNORE_UID 的值，但值必須與您在上一個步驟中指定的值相同；例如 1337。如有必要，您可以新增其他地址到 APPMESH_EGRESS_IGNORED_IP。請勿修改任何其他行。

```
#!/bin/bash -e

#
# Start of configurable options
#


#APPMESH_START_ENABLED="0"
APPMESH_IGNORE_UID="1337"
APPMESH_APP_PORTS="8000"
APPMESH_ENVOY_EGRESS_PORT="15001"
APPMESH_ENVOY_INGRESS_PORT="15000"
APPMESH_EGRESS_IGNORED_IP="169.254.169.254,169.254.170.2"

# Enable routing on the application start.
[ -z "$APPMESH_START_ENABLED" ] && APPMESH_START_ENABLED="0"

# Enable IPv6.
[ -z "$APPMESH_ENABLE_IPV6" ] && APPMESH_ENABLE_IPV6="0"

# Egress traffic from the processess owned by the following UID/GID will be
 ignored.
if [ -z "$APPMESH_IGNORE_UID" ] && [ -z "$APPMESH_IGNORE_GID" ]; then
    echo "Variables APPMESH_IGNORE_UID and/or APPMESH_IGNORE_GID must be set."
    echo "Envoy must run under those IDs to be able to properly route it's egress
 traffic."
    exit 1
fi

# Port numbers Application and Envoy are listening on.
if [ -z "$APPMESH_ENVOY_EGRESS_PORT" ]; then
    echo "APPMESH_ENVOY_EGRESS_PORT must be defined to forward traffic from the
 application to the proxy."
    exit 1
fi

# If an app port was specified, then we also need to enforce the proxies ingress
 port so we know where to forward traffic.
if [ ! -z "$APPMESH_APP_PORTS" ] && [ -z "$APPMESH_ENVOY_INGRESS_PORT" ]; then
```

```
    echo "APPMESH_ENVOY_INGRESS_PORT must be defined to forward traffic from the
 APPMESH_APP_PORTS to the proxy."
    exit 1
fi

# Comma separated list of ports for which egress traffic will be ignored, we always
 refuse to route SSH traffic.
if [ -z "$APPMESH_EGRESS_IGNORED_PORTS" ]; then
    APPMESH_EGRESS_IGNORED_PORTS="22"
else
    APPMESH_EGRESS_IGNORED_PORTS="$APPMESH_EGRESS_IGNORED_PORTS,22"
fi

#
# End of configurable options
#

function initialize() {
    echo "=== Initializing ==="
    if [ ! -z "$APPMESH_APP_PORTS" ]; then
        iptables -t nat -N APPMESH_INGRESS
        if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
            ip6tables -t nat -N APPMESH_INGRESS
        fi
    fi
    iptables -t nat -N APPMESH_EGRESS
    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        ip6tables -t nat -N APPMESH_EGRESS
    fi
}

function enable_egress_routing() {
    # Stuff to ignore
    [ ! -z "$APPMESH_IGNORE_UID" ] && \
        iptables -t nat -A APPMESH_EGRESS \
        -m owner --uid-owner $APPMESH_IGNORE_UID \
        -j RETURN

    [ ! -z "$APPMESH_IGNORE_GID" ] && \
        iptables -t nat -A APPMESH_EGRESS \
        -m owner --gid-owner $APPMESH_IGNORE_GID \
        -j RETURN

    [ ! -z "$APPMESH_EGRESS_IGNORED_PORTS" ] && \
```

```
        for IGNORED_PORT in $(echo "$APPMESH_EGRESS_IGNORED_PORTS" | tr "," "\n");
do
          iptables -t nat -A APPMESH_EGRESS \
          -p tcp \
          -m multiport --dports "$IGNORED_PORT" \
          -j RETURN
        done

   if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
     # Stuff to ignore ipv6
     [ ! -z "$APPMESH_IGNORE_UID" ] && \
         ip6tables -t nat -A APPMESH_EGRESS \
         -m owner --uid-owner $APPMESH_IGNORE_UID \
         -j RETURN

     [ ! -z "$APPMESH_IGNORE_GID" ] && \
         ip6tables -t nat -A APPMESH_EGRESS \
         -m owner --gid-owner $APPMESH_IGNORE_GID \
         -j RETURN

     [ ! -z "$APPMESH_EGRESS_IGNORED_PORTS" ] && \
       for IGNORED_PORT in $(echo "$APPMESH_EGRESS_IGNORED_PORTS" | tr "," "\n");
do
         ip6tables -t nat -A APPMESH_EGRESS \
         -p tcp \
         -m multiport --dports "$IGNORED_PORT" \
         -j RETURN
       done
   fi

   # The list can contain both IPv4 and IPv6 addresses. We will loop over this
list
   # to add every IPv4 address into `iptables` and every IPv6 address into
`ip6tables`.
   [ ! -z "$APPMESH_EGRESS_IGNORED_IP" ] && \
       for IP_ADDR in $(echo "$APPMESH_EGRESS_IGNORED_IP" | tr "," "\n"); do
           if [[ $IP_ADDR =~ .*:.* ]]
           then
               [ "$APPMESH_ENABLE_IPV6" == "1" ] && \
                   ip6tables -t nat -A APPMESH_EGRESS \
                       -p tcp \
                       -d "$IP_ADDR" \
                       -j RETURN
           else
```

```
                iptables -t nat -A APPMESH_EGRESS \
                    -p tcp \
                    -d "$IP_ADDR" \
                    -j RETURN
            fi
        done

    # Redirect everything that is not ignored
    iptables -t nat -A APPMESH_EGRESS \
        -p tcp \
        -j REDIRECT --to $APPMESH_ENVOY_EGRESS_PORT

    # Apply APPMESH_EGRESS chain to non local traffic
    iptables -t nat -A OUTPUT \
        -p tcp \
        -m addrtype ! --dst-type LOCAL \
        -j APPMESH_EGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        # Redirect everything that is not ignored ipv6
        ip6tables -t nat -A APPMESH_EGRESS \
            -p tcp \
            -j REDIRECT --to $APPMESH_ENVOY_EGRESS_PORT
        # Apply APPMESH_EGRESS chain to non local traffic ipv6
        ip6tables -t nat -A OUTPUT \
            -p tcp \
            -m addrtype ! --dst-type LOCAL \
            -j APPMESH_EGRESS
    fi

}

function enable_ingress_redirect_routing() {
    # Route everything arriving at the application port to Envoy
    iptables -t nat -A APPMESH_INGRESS \
        -p tcp \
        -m multiport --dports "$APPMESH_APP_PORTS" \
        -j REDIRECT --to-port "$APPMESH_ENVOY_INGRESS_PORT"

    # Apply AppMesh ingress chain to everything non-local
    iptables -t nat -A PREROUTING \
        -p tcp \
        -m addrtype ! --src-type LOCAL \
        -j APPMESH_INGRESS
```

```
    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        # Route everything arriving at the application port to Envoy ipv6
        ip6tables -t nat -A APPMESH_INGRESS \
            -p tcp \
            -m multiport --dports "$APPMESH_APP_PORTS" \
            -j REDIRECT --to-port "$APPMESH_ENVOY_INGRESS_PORT"

        # Apply AppMesh ingress chain to everything non-local ipv6
        ip6tables -t nat -A PREROUTING \
            -p tcp \
            -m addrtype ! --src-type LOCAL \
            -j APPMESH_INGRESS
    fi
}

function enable_routing() {
    echo "=== Enabling routing ==="
    enable_egress_routing
    if [ ! -z "$APPMESH_APP_PORTS" ]; then
        enable_ingress_redirect_routing
    fi
}

function disable_routing() {
    echo "=== Disabling routing ==="
    iptables -t nat -F APPMESH_INGRESS
    iptables -t nat -F APPMESH_EGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        ip6tables -t nat -F APPMESH_INGRESS
        ip6tables -t nat -F APPMESH_EGRESS
    fi
}

function dump_status() {
    echo "=== iptables FORWARD table ==="
    iptables -L -v -n
    echo "=== iptables NAT table ==="
    iptables -t nat -L -v -n

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        echo "=== ip6tables FORWARD table ==="
        ip6tables -L -v -n
```

```
        echo "=== ip6tables NAT table ==="
        ip6tables -t nat -L -v -n
    fi
}

function clean_up() {
    disable_routing
    ruleNum=$(iptables -L PREROUTING -t nat --line-numbers | grep APPMESH_INGRESS |
 cut -d " " -f 1)
    iptables -t nat -D PREROUTING $ruleNum

    ruleNum=$(iptables -L OUTPUT -t nat --line-numbers | grep APPMESH_EGRESS | cut
 -d " " -f 1)
    iptables -t nat -D OUTPUT $ruleNum

    iptables -t nat -X APPMESH_INGRESS
    iptables -t nat -X APPMESH_EGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        ruleNum=$(ip6tables -L PREROUTING -t nat --line-numbers | grep
 APPMESH_INGRESS | cut -d " " -f 1)
        ip6tables -t nat -D PREROUTING $ruleNum

        ruleNum=$(ip6tables -L OUTPUT -t nat --line-numbers | grep APPMESH_EGRESS |
 cut -d " " -f 1)
        ip6tables -t nat -D OUTPUT $ruleNum

        ip6tables -t nat -X APPMESH_INGRESS
        ip6tables -t nat -X APPMESH_EGRESS
    fi
}

function main_loop() {
    echo "=== Entering main loop ==="
    while read -p '> ' cmd; do
        case "$cmd" in
            "quit")
                clean_up
                break
                ;;
            "status")
                dump_status
                ;;
            "enable")
```

```
                enable_routing
                ;;
            "disable")
                disable_routing
                ;;
            *)
                echo "Available commands: quit, status, enable, disable"
                ;;
        esac
    done
}

function print_config() {
    echo "=== Input configuration ==="
    env | grep APPMESH_ || true
}

print_config

initialize

if [ "$APPMESH_START_ENABLED" == "1" ]; then
    enable_routing
fi

main_loop
```

9. 若要設定 iptables 規則以將應用程式流量路由至 Envoy 代理，請執行您在上一個步驟中建立的指令碼。

```
sudo ./envoy-networking.sh
```

10. 啟動您的虛擬節點應用程式程式碼。

> **Note**
>
> 如需 App Mesh 的更多範例和逐步解說，請參閱 App Mesh 範例儲存庫。

App Mesh 範例

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

您可以在下列儲存庫中找到end-to-end演練 AWS App Mesh ，顯示與各種 AWS 服務整合的動作和程式碼範例：

應用程式網格範例

應用程式網格概念

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

App Mesh 由下列概念組成。

- 服務網格
- 虛擬服務
- 虛擬閘道
- 虛擬節點
- 虛擬路由器

服務網格

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

服務網格是在它之內各服務之間網路流量的邏輯邊界。建立您的服務網格後，您可以建立虛擬服務、虛擬節點、虛擬路由器和路由，以分配網格中的應用程式之間的流量。

建立服務網格

> **Note**
>
> 建立網格時，您必須新增命名空間選擇器。如果命名空間選擇器是空的，則會選取所有命名空間。若要限制命名空間，請使用標籤將 App Mesh 資源與建立的網格建立關聯。

AWS Management Console

使用 建立服務網格 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇 Create mesh (建立網格)。
3. 對於 Mesh name (網格名稱)，為您的服務網格指定名稱。
4. （選用）選擇允許外部流量。在預設情況下，網格中的代理只會互相轉送流量。如果您允許外部流量，則網格中的代理也會將 TCP 流量直接轉送至未透過網格中定義的代理部署的服務。

> **Note**
>
> 如果您在使用 ALLOW_ALL 時在虛擬節點上指定任何後端，您必須指定該虛擬節點的所有輸出作為後端。否則，ALLOW_ALL 將不再適用於該虛擬節點。

5. IP 版本偏好設定

透過切換覆寫預設 IP 版本行為，控制哪些 IP 版本應該用於網格內的流量。根據預設，App Mesh 會使用各種 IP 版本。

> **Note**
>
> 網格會將 IP 偏好設定套用至網格內的所有虛擬節點和虛擬閘道。當您建立或編輯節點時，可透過設定 IP 偏好設定，在個別虛擬節點上覆寫此行為。虛擬閘道上的 IP 偏好設定無法覆寫，因為虛擬閘道的組態允許它們同時接聽 IPv4 和 IPv6 流量，無論網格上設定的偏好設定為何，都相同。

- 預設
  - Envoy 的 DNS 解析程式偏好 IPv6，並回到 IPv4。
  - 如果 AWS Cloud Map 可用，我們會使用 傳回IPv4的地址，並使用該IPv6地址返回 。
  - 為本機應用程式建立的端點會使用 IPv4地址。
  - Envoy 接聽程式會繫結至所有IPv4地址。
- IPv6 偏好
  - Envoy 的 DNS 解析程式偏好 IPv6，並回到 IPv4。
  - 如果可用，則會使用 AWS Cloud Map 傳回IPv6的地址，並回復為使用該IPv4地址

- 為本機應用程式建立的端點使用 IPv6地址。
- Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。

- IPv4 偏好
  - Envoy 的 DNS 解析程式偏好 IPv4，並回到 IPv6。
  - 如果 AWS Cloud Map 可用，我們會使用 傳回IPv4的地址，並使用該IPv6地址返回 。
  - 為本機應用程式建立的端點會使用 IPv4地址。
  - Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。
- 僅限 IPv6
  - Envoy 的 DNS 解析程式僅使用 IPv6。
  - 只會 AWS Cloud Map 使用 傳回IPv6的地址。如果 AWS Cloud Map 傳回IPv4地址，則不會使用 IP 地址，而空白結果會傳回至 Envoy。
  - 為本機應用程式建立的端點使用 IPv6地址。
  - Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。
- 僅限 IPv4
  - Envoy 的 DNS 解析程式僅使用 IPv4。
  - 只會 AWS Cloud Map 使用 傳回IPv4的地址。如果 AWS Cloud Map 傳回IPv6地址，則不會使用 IP 地址，而空白結果會傳回至 Envoy。
  - 為本機應用程式建立的端點會使用 IPv4地址。
  - Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。

6. 選擇 Create mesh (建立網格) 以完成。
7. （選用）與其他帳戶共用網格。共用網格可讓不同帳戶建立的資源在同一個網格中彼此通訊。如需詳細資訊，請參閱使用共用網格。

AWS CLI

使用 建立網格 AWS CLI。

使用下列命令建立服務網格（使用您自己的命令取代*##*值）：

1.
```
aws appmesh create-mesh --mesh-name meshName
```
2. 輸出範例：

```
{
```

```
    "mesh":{
        "meshName":"meshName",
        "metadata":{
            "arn":"arn:aws:appmesh:us-west-2:123456789012:mesh/meshName",
            "createdAt":"2022-04-06T08:45:50.072000-05:00",
            "lastUpdatedAt":"2022-04-06T08:45:50.072000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version":1
        },
        "spec":{},
        "status":{
            "status":"ACTIVE"
        }
    }
}
```

如需使用 AWS CLI for App Mesh 建立網格的詳細資訊，請參閱 AWS CLI 參考中的 create-mesh 命令。

刪除網格

AWS Management Console

使用 刪除虛擬閘道 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇要刪除的網格。會列出您擁有和已與您共用的所有網格。
3. 在確認方塊中，輸入 **delete**，然後按一下刪除。

AWS CLI

使用 刪除網格 AWS CLI

1. 使用下列命令來刪除網格（使用您自己的值取代*##*值）：

```
aws appmesh delete-mesh \
     --mesh-name meshName
```

2. 輸出範例：

```
{
    "mesh": {
        "meshName": "meshName",
        "metadata": {
            "arn":"arn:aws:appmesh:us-west-2:123456789012:mesh/meshName",
            "createdAt": "2022-04-06T08:45:50.072000-05:00",
            "lastUpdatedAt": "2022-04-07T11:06:32.795000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {},
        "status": {
            "status": "DELETED"
        }
    }
}
```

如需使用 AWS CLI for App Mesh 刪除網格的詳細資訊，請參閱 AWS CLI 參考中的 delete-mesh 命令。

虛擬服務

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

虛擬服務是實際服務的抽象化，由虛擬節點直接提供，或透過虛擬路由器間接提供。相依服務會以 `virtualServiceName` 呼叫您的虛擬服務，而且這些請求會路由傳送到虛擬節點或虛擬路由器 (指定為虛擬服務的提供者)。

建立虛擬服務

AWS Management Console

使用 建立虛擬服務 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要在其中建立虛擬服務的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual services (虛擬服務)。
4. 選擇 Create virtual service (建立虛擬服務)。
5. 對於 Virtual service name (虛擬服務名稱)，為您的虛擬服務選擇名稱。
   您可以選擇任何名稱，但建議您目標為 等實際服務的服務探索名稱`my-service.default.svc.cluster.local`，可讓您更輕鬆地將虛擬服務與實際服務建立關聯。如此一來，您不需要變更程式碼來參考與目前程式碼不同的名稱。您指定的名稱必須解析為非回溯 IP 地址，因為應用程式容器必須能夠成功解析名稱，才能將請求傳送至 Envoy 代理。您可以使用任何非回溯 IP 地址，因為應用程式或代理容器都無法與此 IP 地址通訊。代理會透過您在 App Mesh 中為它們設定的名稱與其他虛擬服務通訊，而不是透過名稱解析的 IP 地址。
6. 針對 Provider (提供者)，為您的虛擬服務選擇提供者類型：
   - 如果您希望虛擬服務將流量分散到多個虛擬節點，請選取 Virtual router (虛擬路由器)，然後從下拉式功能表選擇要使用的虛擬路由器。
   - 如果您希望虛擬服務在沒有虛擬路由器的情況下直接到達虛擬節點，請選取虛擬節點，然後從下拉式功能表中選擇要使用的虛擬節點。

     > **Note**
     >
     > App Mesh 可能會為您在 2020 年 7 月 29 日當天或之後定義的每個虛擬節點提供者自動建立預設 Envoy 路由重試政策，即使您無法透過 App Mesh API 定義此類政策。如需詳細資訊，請參閱預設路由重試政策。

   - 如果您此時不想讓虛擬服務路由傳送流量 (例如，如果您的虛擬節點或虛擬路由器尚不存在)，請選擇 None (無)。您稍後可以更新此虛擬服務的提供者。
7. 選擇 Create virtual service (建立虛擬服務) 以完成。

AWS CLI

使用 建立虛擬服務 AWS CLI。

使用下列命令和輸入 JSON 檔案，使用虛擬節點提供者建立虛擬服務 （使用您自己的值取代## #值）：

1.
```
aws appmesh create-virtual-service \
--cli-input-json file://create-virtual-service-virtual-node.json
```

2. create-virtual-service-virtual-node.json 範例的內容：

```
{
    "meshName": "meshName",
    "spec": {
        "provider": {
            "virtualNode": {
                "virtualNodeName": "nodeName"
            }
        }
    },
    "virtualServiceName": "serviceA.svc.cluster.local"
}
```

3. 輸出範例：

```
{
    "virtualService": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualService/serviceA.svc.cluster.local",
            "createdAt": "2022-04-06T09:45:35.890000-05:00",
            "lastUpdatedAt": "2022-04-06T09:45:35.890000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "provider": {
                "virtualNode": {
                    "virtualNodeName": "nodeName"
```

```
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualServiceName": "serviceA.svc.cluster.local"
    }
}
```

如需使用 AWS CLI for App Mesh 建立虛擬服務的詳細資訊，請參閱 AWS CLI 參考中的 create-virtual-service 命令。

刪除虛擬服務

> **Note**
>
> 您無法刪除閘道路由參考的虛擬服務。您需要先刪除閘道路由。

AWS Management Console

使用 刪除虛擬服務 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您想要從中刪除虛擬服務的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual services (虛擬服務)。
4. 選擇您要刪除的虛擬服務，然後按一下右上角的刪除。您只能刪除將帳戶列為資源擁有者的虛擬閘道。
5. 在確認方塊中，輸入 **`delete`**，然後按一下刪除。

AWS CLI

使用 刪除虛擬服務 AWS CLI

1. 使用下列命令來刪除您的虛擬服務 （ 將##值取代為您自己的值 ）：

```
aws appmesh delete-virtual-service \
     --mesh-name meshName \
     --virtual-service-name serviceA.svc.cluster.local
```

2. 輸出範例：

```
{
    "virtualService": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualService/serviceA.svc.cluster.local",
            "createdAt": "2022-04-06T09:45:35.890000-05:00",
            "lastUpdatedAt": "2022-04-07T10:39:42.772000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "provider": {
                "virtualNode": {
                    "virtualNodeName": "nodeName"
                }
            }
        },
        "status": {
            "status": "DELETED"
        },
        "virtualServiceName": "serviceA.svc.cluster.local"
    }
}
```

如需使用 AWS CLI for App Mesh 刪除虛擬服務的詳細資訊，請參閱 AWS CLI 參考中的 delete-virtual-service 命令。

虛擬閘道

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

虛擬閘道可讓網格外的資源與網格內的資源進行通訊。虛擬閘道代表在 Amazon ECS 服務、Kubernetes 服務或 Amazon EC2 執行個體上執行的 Envoy 代理。與代表使用應用程式執行之 Envoy 的虛擬節點不同，虛擬閘道代表 Envoy 本身部署。

外部資源必須能夠將 DNS 名稱解析為指派給執行 Envoy 之服務或執行個體的 IP 地址。然後，Envoy 可以存取網格內資源的所有 App Mesh 組態。在虛擬閘道處理傳入請求的組態會使用閘道路由指定。

> **Important**
>
> 具有 HTTP 或 HTTP2 接聽程式的虛擬閘道會將傳入請求的主機名稱重寫為 Gateway Route 目標 Virtual Service 的名稱，且閘道路由的相符字首/預設為重寫至 。例如，如果您已將閘道路由比對字首設定為 ，`/chapter`且傳入請求為 `/chapter/1`，則請求會重寫至 `/1`。若要設定重寫，請參閱從閘道路由建立閘道路由一節。
> 建立虛擬閘道時，`user`不應設定 `proxyConfiguration`和 。

若要完成end-to-end演練，請參閱設定傳入閘道。

建立虛擬閘道

> **Note**
>
> 建立虛擬閘道時，您必須新增具有標籤的命名空間選擇器，以識別要將閘道路由與建立的虛擬閘道建立關聯的命名空間清單。

AWS Management Console

使用 建立虛擬閘道 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要在其中建立虛擬閘道的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中選擇虛擬閘道。
4. 選擇建立虛擬閘道。
5. 針對虛擬閘道名稱，輸入虛擬閘道的名稱。
6. （選用，但建議使用）設定用戶端政策預設值。

   a. （選用）如果您希望閘道只使用 Transport Layer Security (TLS) 與虛擬服務通訊，請選取強制執行 TLS。

   b. （選用）針對連接埠，指定您要在其中強制與虛擬服務進行 TLS 通訊的一或多個連接埠。

   c. 針對驗證方法，選取下列其中一個選項。您指定的憑證必須已存在且符合特定要求。如需詳細資訊，請參閱憑證需求。

      - AWS Private Certificate Authority 託管：選取一或多個現有的憑證。
      - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 使用 Secret Discovery Service 擷取的秘密名稱。
      - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑。

   d. （選用）輸入主體別名。若要新增其他 SANs，請選取新增 SAN。SANs必須為 FQDN 或 URI 格式。

   e. （選用）選取提供用戶端憑證和下列其中一個選項，以在伺服器請求時提供用戶端憑證，並啟用相互 TLS 身分驗證。若要進一步了解相互 TLS，請參閱應用程式網格相互 TLS 身分驗證文件。

      - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 使用 Secret Discovery Service 擷取的秘密名稱。
      - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑，以及私有金鑰。如需使用本機檔案加密，以範例應用程式部署網格的完整end-to-end逐步解說，請參閱在 GitHub 上設定 TLS 與檔案提供的 TLS 憑證。

7. （選用）若要設定記錄，請選取記錄。輸入您希望 Envoy 使用的 HTTP 存取日誌路徑。建議您使用 `/dev/stdout` 路徑，以便您可以使用 Docker 日誌驅動程式將您的 Envoy 日誌匯出至 Amazon CloudWatch Logs 等服務。

> **Note**
>
> 日誌必須仍然由您的應用程式中的代理程式輸入，並傳送到目的地。這個檔案路徑只是指示 Envoy 將日誌傳送到何處。

8. 設定接聽程式。

   a. 選取通訊協定，並指定 Envoy 接聽流量的連接埠。http 接聽程式允許連線轉換至 Websocket。您可以按一下新增接聽程式來新增多個接聽程式。移除按鈕會移除該接聽程式。

   b. （選用）啟用連線集區

      連線集區會限制 Virtual Gateway Envoy 可同時建立的連線數量。其旨在保護您的 Envoy 執行個體免於因連線而負擔過重，並可讓您根據應用程式的需求調整流量調整。

      您可以設定虛擬閘道接聽程式的目的地連線集區設定。App Mesh 預設會將用戶端連線集區設定設定為無限，簡化網格組態。

      > **Note**
      >
      > `connectionPool` 和 `connectionPoolportMapping` 通訊協定必須相同。
      > 如果您的接聽程式通訊協定是 `grpc`或 `http2`，請`maxRequests`僅指定 。
      > 如果您的接聽程式通訊協定是 `http`，您可以同時指定 `maxConnections`和 `maxPendingRequests`。

      - 對於最大連線數，請指定最大輸出連線數。
      - 針對請求上限，指定可使用 Virtual Gateway Envoy 建立的平行請求數量上限。
      - （選用）針對最大待處理請求，指定 Envoy 佇列的最大連線數之後的溢位請求數。預設值為 2147483647。

   c. （選用）如果您想要為接聽程式設定運作狀態檢查，請選取啟用運作狀態檢查。

      運作狀態檢查政策是選用的，但如果您指定運作狀態政策的任何值，則必須指定運作狀態閾值、運作狀態檢查間隔、運作狀態檢查通訊協定、逾時期間和運作狀態不佳閾值的值。

      - 針對運作狀態檢查通訊協定，選擇通訊協定。如果您選擇 grpc，則您的服務必須符合 GRPC 運作狀態檢查通訊協定。

- 對於 Health check port (運作狀態檢查連接埠)，指定應執行運作狀態檢查的連接埠。
- 對於 Healthy threshold (運作良好閾值)，指定在宣告接聽程式運作良好之前，必須達到的運作狀態檢查連續成功次數。
- 對於 Health check interval (運作狀態檢查間隔)，指定每次運作狀態檢查執行之間的時間間隔 (以毫秒為單位)。
- 對於 Path (路徑)，指定運作狀態檢查請求的目的地路徑。只有在運作狀態檢查通訊協定為 `http`或 時，才會使用此值`http2`。其他通訊協定會忽略此值。
- 針對逾時期間，指定從運作狀態檢查收到回應時所等待的時間量，以毫秒為單位。
- 對於 Unhealthy threshold (運作不良閾值)，指定在宣告接聽程式運作不良之前，必須達到的運作狀態檢查連續失敗次數。

d. （選用）如果您想要指定用戶端是否使用 TLS 與此虛擬閘道通訊，請選取啟用 TLS 終止。

- 針對 模式，選取您想要在接聽程式上設定 TLS 的模式。
- 針對憑證方法，選取下列其中一個選項。憑證必須符合特定要求。如需詳細資訊，請參閱憑證需求。
  - AWS Certificate Manager 託管 – 選取現有的憑證。
  - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 使用 Secret Discovery Service 擷取的秘密名稱。
  - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈和私有金鑰檔案的路徑。
- （選用）選取需要用戶端憑證和下列其中一個選項，以便在用戶端提供憑證時啟用相互 TLS 身分驗證。若要進一步了解相互 TLS，請參閱應用程式網格相互 TLS 身分驗證文件。
  - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 使用 Secret Discovery Service 擷取的秘密名稱。
  - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑。
- （選用）輸入主體別名。若要新增其他 SANs，請選取新增 SAN。SANs必須為 FQDN 或 URI 格式。

9. 選擇建立虛擬閘道以完成。

AWS CLI

使用 建立虛擬閘道 AWS CLI。

使用下列命令和輸入 JSON 建立虛擬閘道（使用您自己的值取代*##*值）：

1.
```
aws appmesh create-virtual-gateway \
--mesh-name meshName \
--virtual-gateway-name virtualGatewayName \
--cli-input-json file://create-virtual-gateway.json
```

2. create-virtual-gateway.json 範例的內容：

```
{
    "spec": {
      "listeners": [
        {
          "portMapping": {
            "port": 9080,
            "protocol": "http"
          }
        }
      ]
    }
}
```

3. 輸出範例：

```
{
    "virtualGateway": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/
virtualGateway/virtualGatewayName",
            "createdAt": "2022-04-06T10:42:42.015000-05:00",
            "lastUpdatedAt": "2022-04-06T10:42:42.015000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 9080,
                        "protocol": "http"
```

```
                    }
                }
            ]
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualGatewayName": "virtualGatewayName"
    }
}
```

如需使用 AWS CLI for App Mesh 建立虛擬閘道的詳細資訊，請參閱 AWS CLI 參考中的 create-virtual-gateway 命令。

部署虛擬閘道

部署僅包含 Envoy 容器的 Amazon ECS 或 Kubernetes 服務。您也可以在 Amazon EC2 執行個體上部署 Envoy 容器。如需詳細資訊，請參閱開始使用 App Mesh 和 Amazon EC2。如需如何在 Amazon ECS 上部署的詳細資訊，請參閱開始使用 App Mesh 和 Amazon ECS 或開始使用 AWS App Mesh 和 Kubernetes 以部署至 Kubernetes。您需要將APPMESH_RESOURCE_ARN環境變數設定為 ，mesh/*mesh-name*/virtualGateway/*virtual-gateway-name*且不得指定代理組態，這樣代理的流量就不會重新導向至本身。根據預設，當 Envoy 在指標和追蹤方面參照本身時，App Mesh 會使用您在 APPMESH_RESOURCE_ARN 中指定的資源名稱。您可以藉由使用自己的名稱設定 APPMESH_RESOURCE_CLUSTER 環境變數，以覆寫此行為。

我們建議您部署多個容器執行個體，並設定 Network Load Balancer 以將流量負載平衡至執行個體。負載平衡器的服務探索名稱是您希望外部服務用來存取網格中資源的名稱，例如 *myapp.example.com*。如需詳細資訊，請參閱建立 Network Load Balancer (Amazon ECS)、建立 External Load Balancer (Kubernetes) 或教學課程：提高 Amazon EC2 上應用程式的可用性。您也可以在我們的 App Mesh 範例中找到更多範例和演練。

啟用 Envoy 的代理授權。如需詳細資訊，請參閱Envoy Proxy 授權。

刪除虛擬閘道

AWS Management Console

使用 刪除虛擬閘道 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您想要從中刪除虛擬閘道的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中選擇虛擬閘道。
4. 選擇您要刪除的虛擬閘道，然後選擇刪除。如果虛擬閘道具有任何相關聯的閘道路由，則無法刪除它。您必須先刪除任何相關聯的閘道路由。您只能刪除您帳戶列為資源擁有者的虛擬閘道。
5. 在確認方塊中，輸入 **delete**，然後選取刪除。

AWS CLI

使用 刪除虛擬閘道 AWS CLI

1. 使用下列命令來刪除您的虛擬閘道 （ 將*##*值取代為您自己的值 ）：

```
aws appmesh delete-virtual-gateway \
     --mesh-name meshName \
     --virtual-gateway-name virtualGatewayName
```

2. 輸出範例：

```
{
    "virtualGateway": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/
virtualGateway/virtualGatewayName",
            "createdAt": "2022-04-06T10:42:42.015000-05:00",
            "lastUpdatedAt": "2022-04-07T10:57:22.638000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
```

```
            "listeners": [
                {
                    "portMapping": {
                        "port": 9080,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "DELETED"
        },
        "virtualGatewayName": "virtualGatewayName"
    }
}
```

如需使用 AWS CLI for App Mesh 刪除虛擬閘道的詳細資訊，請參閱 AWS CLI 參考中的 delete-virtual-gateway 命令。

閘道路由

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

閘道路由會連結至虛擬閘道，並將流量路由傳送至現有的虛擬服務。如果路由符合請求，它可以將流量分配到目標虛擬服務。本主題可協助您在服務網格中使用閘道路由。

建立閘道路由

AWS Management Console

使用 建立閘道路由 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要在其中建立閘道路由的網格。會列出您擁有和已與您共用的所有網格。

3. 在左側導覽中選擇虛擬閘道。
4. 選擇您要與新閘道路由建立關聯的虛擬閘道。如果沒有列出，則需要先建立虛擬閘道。您只能為將帳戶列為資源擁有者的虛擬閘道建立閘道路由。
5. 在閘道路由表中，選擇建立閘道路由。
6. 針對閘道路由名稱，指定要用於閘道路由的名稱。
7. 針對閘道路由類型，選擇 http、http2 或 grpc。
8. 選取現有的虛擬服務名稱。如果沒有列出，則需要先建立虛擬服務。
9. 選擇與虛擬服務供應商連接埠目標對應的連接埠。當所選虛擬服務的提供者（路由器或節點）有多個接聽程式時，需要虛擬服務提供者連接埠。
10. （選用）針對優先順序，指定此閘道路由的優先順序。
11. 針對相符組態，指定：

    - 如果 http/http2 是選取的類型：
      - （選用）方法 - 指定要在傳入 http/http2 請求中比對的方法標頭。
      - （選用）連接埠比對 - 比對傳入流量的連接埠。如果此虛擬閘道有多個接聽程式，則需要連接埠比對。
      - （選用）確切/尾碼主機名稱 - 指定應該在傳入請求上比對的主機名稱，以路由至目標虛擬服務。
      - （選用）Prefix/Exact/Regex 路徑 - 符合 URL 路徑的方法。
        - 字首比對 - 根據預設/，閘道路由相符的請求會重寫至目標虛擬服務的名稱，相符的字首則會重寫至 。視您設定虛擬服務的方式而定，它可以使用虛擬路由器，根據特定字首或標頭，將請求路由到不同的虛擬節點。

      > **Important**
      >
      > - 您無法/aws-app-mesh*為字首比對指定 /aws-appmesh*或 。這些字首保留供未來 App Mesh 內部使用。
      > - 如果定義了多個閘道路由，則請求會與字首最長的路由相符。例如，如果存在兩個閘道路由，其中一個具有 字首，/chapter另一個具有 字首/，則 的請求www.example.com/chapter/會與具有 /chapter字首的閘道路由相符。

> **Note**
>
> 如果您啟用以路徑/字首為基礎的比對，App Mesh 會啟用路徑標準化 (normalize_path 和 merge_slashes)，以將路徑混淆漏洞的可能性降至最低。當參與請求的各方使用不同的路徑表示法時，會發生路徑混淆漏洞。

- 完全相符 - 精確參數會停用路由的部分相符，並確保只有在路徑符合目前 URL 時才傳回路由。
- Regex 比對 - 用於描述多個 URLs可能實際識別網站上的單一頁面的模式。

- （選用）查詢參數 - 此欄位可讓您比對查詢參數。
- （選用）標頭 - 指定 http 和 http2 的標頭。它應該符合傳入請求，以路由到目標虛擬服務。

- 如果 grpc 是選取的類型：
  - 主機名稱比對類型和（選用）精確/尾碼比對 - 指定應該在傳入請求上比對的主機名稱，以路由至目標虛擬服務。
  - grpc 服務名稱 - grpc 服務可做為應用程式的 API，並使用 ProtoBuf 定義。

> **Important**
>
> 您無法`aws.appmesh`為服務名稱指定 `/aws.app-mesh*`或 。這些服務名稱保留供未來的 App Mesh 內部使用。

  - （選用）中繼資料 - 指定 grpc 的中繼資料。它應該符合路由至目標虛擬服務的傳入請求。

12. （選用）針對重寫組態：

- 如果 http/http2 是選取的類型：
  - 如果字首是選取的比對類型：
    - 覆寫主機名稱的自動重寫 - 根據預設，主機名稱會重寫為目標虛擬服務的名稱。
    - 覆寫字首的自動重寫 - 開啟時，字首重寫會指定重寫字首的值。
  - 如果精確路徑是選取的比對類型：
    - 覆寫主機名稱的自動重寫 - 預設會將主機名稱重寫為目標虛擬服務的名稱。
    - 路徑重寫 - 指定重寫路徑的值。沒有預設路徑。

- 如果 Regex 路徑是選取的比對類型：
  - 覆寫主機名稱的自動重寫 - 預設會將主機名稱重寫為目標虛擬服務的名稱。
  - 路徑重寫 - 指定重寫路徑的值。沒有預設路徑。
- 如果 grpc 是選取的類型：
  - 覆寫主機名稱的自動重寫 - 根據預設，主機名稱會重寫為目標虛擬服務的名稱。

13. 選擇建立閘道路由以完成。

AWS CLI

使用 建立閘道路由 AWS CLI。

使用下列命令建立閘道路由並輸入 JSON （使用您自己的值取代*##*值）：

1.
```
aws appmesh create-virtual-gateway \
--mesh-name meshName \
--virtual-gateway-name virtualGatewayName \
--gateway-route-name gatewayRouteName \
--cli-input-json file://create-gateway-route.json
```

2. create-gateway-route.json 範例的內容：

```
{
    "spec": {
        "httpRoute" : {
            "match" : {
                "prefix" : "/"
            },
            "action" : {
                "target" : {
                    "virtualService": {
                        "virtualServiceName": "serviceA.svc.cluster.local"
                    }
                }
            }
        }
    }
}
```

3. 輸出範例：

```
{
    "gatewayRoute": {
        "gatewayRouteName": "gatewayRouteName",
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualGateway/virtualGatewayName/gatewayRoute/gatewayRouteName",
            "createdAt": "2022-04-06T11:05:32.100000-05:00",
            "lastUpdatedAt": "2022-04-06T11:05:32.100000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "httpRoute": {
                "action": {
                    "target": {
                        "virtualService": {
                            "virtualServiceName": "serviceA.svc.cluster.local"
                        }
                    }
                },
                "match": {
                    "prefix": "/"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualGatewayName": "gatewayName"
    }
}
```

如需使用 AWS CLI for App Mesh 建立閘道路由的詳細資訊，請參閱 AWS CLI 參考中的 create-gateway-route 命令。

刪除閘道路由

AWS Management Console

使用 刪除閘道路由 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要從中刪除閘道路由的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中選擇虛擬閘道。
4. 選擇您想要從中刪除閘道路由的虛擬閘道。
5. 在閘道路由表中，選擇您要刪除的閘道路由，然後選擇刪除。只有在您的帳戶列為資源擁有者時，您才能刪除閘道路由。
6. 在確認方塊中，輸入 **delete**，然後按一下刪除。

AWS CLI

使用 刪除閘道路由 AWS CLI

1. 使用下列命令來刪除閘道路由 （使用您自己的值取代*##*值）：

```
aws appmesh delete-gateway-route \
     --mesh-name meshName \
     --virtual-gateway-name virtualGatewayName \
     --gateway-route-name gatewayRouteName
```

2. 輸出範例：

```
{
    "gatewayRoute": {
        "gatewayRouteName": "gatewayRouteName",
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualGateway/virtualGatewayName/gatewayRoute/gatewayRouteName",
            "createdAt": "2022-04-06T11:05:32.100000-05:00",
            "lastUpdatedAt": "2022-04-07T10:36:33.191000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
```

```
        },
        "spec": {
            "httpRoute": {
                "action": {
                    "target": {
                        "virtualService": {
                            "virtualServiceName": "serviceA.svc.cluster.local"
                        }
                    }
                },
                "match": {
                    "prefix": "/"
                }
            }
        },
        "status": {
            "status": "DELETED"
        },
        "virtualGatewayName": "virtualGatewayName"
    }
}
```

如需使用 AWS CLI for App Mesh 刪除閘道路由的詳細資訊，請參閱 AWS CLI 參考中的 delete-gateway-route 命令。

虛擬節點

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

虛擬節點扮演特定任務群組的邏輯指標，例如 Amazon ECS 服務或 Kubernetes 部署。建立虛擬節點時，您必須為任務群組指定服務探索方法。虛擬節點預期的任何傳入流量都會指定為接聽程式。虛擬節點傳送傳出流量的任何虛擬服務都會指定為後端。

新虛擬節點的回應中繼資料包含與虛擬節點相關聯的 Amazon Resource Name (ARN)。在 Amazon ECS 任務定義或 Kubernetes Pod 規格中，將此值設定為任務群組 Envoy 代理容器APPMESH_RESOURCE_ARN的環境變數。例如， 值可以是 `arn:aws:appmesh:us-west-2:111122223333:mesh/myMesh/virtualNode/myVirtualNode`。它接著會映射到 `node.id` 和 `node.cluster` Envoy 參數。設定此變數時，您必須使用 Envoy 映像的 `1.15.0`或更新版本。如需 App Mesh Envoy 變數的詳細資訊，請參閱Envoy。

> **Note**
>
> 根據預設，當 Envoy 在指標和追蹤方面參照本身時，App Mesh 會使用您在 APPMESH_RESOURCE_ARN 中指定的資源名稱。您可以藉由使用自己的名稱設定 APPMESH_RESOURCE_CLUSTER 環境變數，以覆寫此行為。

建立虛擬節點

AWS Management Console

使用 建立虛擬節點 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要在其中建立虛擬節點的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual nodes (虛擬節點)。
4. 選擇建立虛擬節點，然後指定虛擬節點的設定。
5. 針對虛擬節點名稱，輸入虛擬節點的名稱。
6. 針對服務探索方法，選擇下列其中一個選項：

   - DNS – 指定虛擬節點代表的實際服務的 DNS 主機名稱。Envoy 代理部署在 Amazon VPC 中。代理會將名稱解析請求傳送至為 VPC 設定的 DNS 伺服器。如果主機名稱解析，DNS 伺服器會傳回一或多個 IP 地址。如需 VPC DNS 設定的詳細資訊，請參閱搭配 VPC 使用 DNS。針對 DNS 回應類型 （選用），指定 DNS 解析程式傳回的端點類型。Load Balancer 表示 DNS 解析程式會傳回一組負載平衡的端點。端點表示 DNS 解析程式正在傳回所有端點。根據預設，回應類型會假設為 Load Balancer。

     > **Note**
     >
     > 如果您使用 Route53，則需要使用 Load Balancer。

- AWS Cloud Map – 指定現有的服務名稱和 HTTP 命名空間。您也可以選擇新增資料列並指定索引鍵和值，以指定 App Mesh 可以 AWS Cloud Map 查詢的屬性。僅傳回符合所有指定鍵/值對的執行個體。若要使用 AWS Cloud Map，您的帳戶必須具有AWSServiceRoleForAppMesh服務連結角色。如需詳細資訊 AWS Cloud Map，請參閱AWS Cloud Map 開發人員指南。
- 無 – 如果您的虛擬節點未預期任何傳入流量，請選取 。

7. IP 版本偏好設定

透過切換覆寫預設 IP 版本行為，控制哪些 IP 版本應該用於網格內的流量。根據預設，App Mesh 會使用各種 IP 版本。

> **Note**
>
> 在虛擬節點上設定 IP 偏好設定只會覆寫針對此特定節點上網格設定的 IP 偏好設定。

- 預設
  - Envoy 的 DNS 解析程式偏好 IPv6，並回到 IPv4。
  - 如果 AWS Cloud Map 可用，我們會使用 傳回IPv4的地址，並回復為使用該IPv6地址。
  - 為本機應用程式建立的端點會使用 IPv4地址。
  - Envoy 接聽程式會繫結至所有IPv4地址。
- IPv6 偏好
  - Envoy 的 DNS 解析程式偏好 IPv6，並回到 IPv4。
  - 如果可用 AWS Cloud Map ，則會使用 傳回IPv6的地址，並回復為使用該IPv4地址
  - 為本機應用程式建立的端點使用 IPv6地址。
  - Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。
- IPv4 偏好
  - Envoy 的 DNS 解析程式偏好 IPv4，並回到 IPv6。
  - 如果 AWS Cloud Map 可用，我們會使用 傳回IPv4的地址，並回復為使用該IPv6地址。
  - 為本機應用程式建立的端點會使用 IPv4地址。
  - Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。
- 僅限 IPv6
  - Envoy 的 DNS 解析程式僅使用 IPv6。

- 只會 AWS Cloud Map 使用 傳回IPv6的地址。如果 AWS Cloud Map 傳回IPv4地址，則不會使用 IP 地址，而空白結果會傳回至 Envoy。
- 為本機應用程式建立的端點使用 IPv6地址。
- Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。

- 僅限 IPv4
  - Envoy 的 DNS 解析程式僅使用 IPv4。
  - 只會 AWS Cloud Map 使用 傳回IPv4的地址。如果 AWS Cloud Map 傳回IPv6地址，則不會使用 IP 地址，而空白結果會傳回至 Envoy。
  - 為本機應用程式建立的端點會使用 IPv4地址。
  - Envoy 接聽程式會繫結至所有 IPv4和 IPv6 地址。

8. （選用）用戶端政策預設值 – 在與後端虛擬服務通訊時設定預設要求。

> **Note**
>
> - 如果您想要為現有的虛擬節點啟用 Transport Layer Security (TLS)，建議您建立新的虛擬節點，其代表與現有虛擬節點相同的服務，以啟用 TLS。然後使用虛擬路由器和路由，逐漸將流量轉移到新的虛擬節點。如需建立路由和調整轉換權重的詳細資訊，請參閱 路由。如果您使用 TLS 更新現有的流量服務虛擬節點，下游用戶端 Envoy 代理可能會先收到 TLS 驗證內容，然後您更新之虛擬節點的 Envoy 代理才會收到憑證。這可能會導致下游 Envoy 代理發生 TLS 交涉錯誤。
> - 必須為部署的 Envoy 代理啟用代理授權，其應用程式由後端服務的虛擬節點表示。建議您啟用代理授權時，只能存取此虛擬節點正在通訊的虛擬節點。

- （選用）如果您想要要求虛擬節點使用 Transport Layer Security (TLS) 與所有後端通訊，請選取強制執行 TLS。
- （選用）如果您只想要對一或多個特定連接埠使用 TLS，請在連接埠中輸入數字。若要新增其他連接埠，請選取新增連接埠。如果您未指定任何連接埠，則會對所有連接埠強制執行 TLS。
- 針對驗證方法，選取下列其中一個選項。您指定的憑證必須已存在且符合特定要求。如需詳細資訊，請參閱憑證需求。
  - AWS Private Certificate Authority 託管：選取一或多個現有的憑證。如需使用 ACM 憑證加密，以範例應用程式部署網格的完整end-to-end逐步解說，請參閱在 GitHub 上使用 AWS Certificate Manager 設定 TLS。 GitHub

- Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 將使用 Secret Discovery Service 擷取的秘密名稱。
- 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑。如需使用本機檔案加密，以範例應用程式部署網格的完整end-to-end逐步解說，請參閱在 GitHub 上使用檔案提供的 TLS 憑證設定 TLS。 GitHub

- （選用）輸入主體別名。若要新增其他 SANs，請選取新增 SAN。SANs必須為 FQDN 或 URI 格式。
- （選用）選取提供用戶端憑證和下列其中一個選項，以在伺服器請求時提供用戶端憑證，並啟用相互 TLS 身分驗證 。若要進一步了解相互 TLS，請參閱應用程式網格相互 TLS 身分驗證文件。
  - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 將使用 Secret Discovery Service 擷取的秘密名稱。
  - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑，以及私有金鑰。

9. （選用）服務後端 – 指定虛擬節點將與之通訊的 App Mesh 虛擬服務。

- 輸入您虛擬節點通訊之虛擬服務的 App Mesh 虛擬服務名稱或完整 Amazon Resource Name (ARN)。
- （選用）如果您要為後端設定唯一的 TLS 設定，請選取 TLS 設定，然後選取覆寫預設值。
  - （選用）如果您想要虛擬節點使用 TLS 與所有後端通訊，請選取強制執行 TLS。
  - （選用）如果您只想要求對一或多個特定連接埠使用 TLS，請在連接埠中輸入數字。若要新增其他連接埠，請選取新增連接埠。如果您未指定任何連接埠，則會對所有連接埠強制執行 TLS。
  - 針對驗證方法，選取下列其中一個選項。您指定的憑證必須已存在且符合特定要求。如需詳細資訊，請參閱憑證需求。
    - AWS Private Certificate Authority 託管：選取一或多個現有的憑證。
    - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 使用 Secret Discovery Service 擷取的秘密名稱。
    - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑。
  - （選用）輸入主體別名。若要新增其他 SANs，請選取新增 SAN。SANs必須為 FQDN 或 URI 格式。
  - （選用）選取提供用戶端憑證和下列其中一個選項，以在伺服器請求時提供用戶端憑證，並啟用相互 TLS 身分驗證。若要進一步了解相互 TLS，請參閱應用程式網格相互 TLS 身分驗證文件。

- Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 將使用 Secret Discovery Service 擷取的秘密名稱。
- 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑，以及私有金鑰。

- 若要新增其他後端，請選取新增後端。

10. （選用）記錄

    若要設定日誌記錄，請輸入您希望 Envoy 使用的 HTTP 存取日誌路徑。建議您使用 `/dev/stdout` 路徑，以便您可以使用 Docker 日誌驅動程式將您的 Envoy 日誌匯出至 Amazon CloudWatch Logs 等服務。

    > **Note**
    >
    > 日誌必須仍然由您的應用程式中的代理程式輸入，並傳送到目的地。這個檔案路徑只是指示 Envoy 將日誌傳送到何處。

11. 接聽程式組態

    接聽程式支援 HTTPHTTP/2、GRPC、 和 TCP通訊協定。 HTTPS 不支援 。

    a. 如果您的虛擬節點預期傳入流量，請為接聽程式指定連接埠和通訊協定。http 接聽程式允許連線轉換至 Websocket。您可以按一下新增接聽程式來新增多個接聽程式。移除按鈕會移除該接聽程式。

    b. （選用）啟用連線集區

       連線集區會限制 Envoy 可與本機應用程式叢集同時建立的連線數量。它旨在保護本機應用程式免於因連線而負擔過重，並可讓您根據應用程式的需求調整流量形狀。

       您可以設定虛擬節點接聽程式的目的地連線集區設定。App Mesh 預設會將用戶端連線集區設定設定為無限，簡化網格組態。

       > **Note**
       >
       > connectionPool 和 portMapping 通訊協定必須相同。如果您的接聽程式通訊協定是 tcp，請僅指定 maxConnections。如果您的接聽程式通訊協定是 grpc 或 http2，請僅指定 maxRequests。如果您的接聽程式通訊協定是 http，您可以同時指定 maxConnections 和 maxPendingRequests。

- 對於最大連線數，請指定最大輸出連線數。
- （選用）針對最高待處理請求，指定 Envoy 將佇列的最大連線數之後的溢位請求數。預設值為 2147483647。

c. （選用）啟用極端值偵測

在用戶端 Envoy 套用的異常值偵測，可讓用戶端對觀察到的已知錯誤故障連線採取近乎立即的動作。它是一種斷路器實作形式，可追蹤上游服務中個別主機的運作狀態。

異常值偵測會動態判斷上游叢集中的端點是否執行不同於其他端點，並從運作狀態良好的負載平衡集中移除它們。

> **Note**
>
> 若要有效設定伺服器虛擬節點的異常值偵測，該虛擬節點的服務探索方法可以是 AWS Cloud Map 或 DNS，且回應類型欄位設定為 `ENDPOINTS`。如果您使用回應類型為 的 DNS 服務探索方法`LOADBALANCER`，Envoy 代理只會選擇單一 IP 地址來路由到上游服務。這會使從一組主機中退出運作狀態不佳主機的極端值偵測行為失效。如需 Envoy 代理行為與服務探索類型相關的詳細資訊，請參閱服務探索方法一節。

- 對於伺服器錯誤，請指定退出所需的連續 5xx 錯誤數。
- 對於異常值偵測間隔，指定射出掃描分析之間的時間間隔和單位。
- 對於基本退出持續時間，請指定主機退出的基本時間和單位量。
- 針對退出百分比，指定可在負載平衡集區中退出的主機百分比上限。

d. （選用）啟用運作狀態檢查 – 設定運作狀態檢查政策的設定。

運作狀態檢查政策是選用的，但如果您指定運作狀態政策的任何值，則必須指定運作狀態閾值、運作狀態檢查間隔、運作狀態檢查通訊協定、逾時期間和運作狀態不佳閾值的值。

- 針對運作狀態檢查通訊協定，選擇通訊協定。如果您選擇 grpc，則您的服務必須符合 GRPC 運作狀態檢查通訊協定。
- 對於 Health check port (運作狀態檢查連接埠)，指定應執行運作狀態檢查的連接埠。
- 對於 Healthy threshold (運作良好閾值)，指定在宣告接聽程式運作良好之前，必須達到的運作狀態檢查連續成功次數。

- 對於 Health check interval (運作狀態檢查間隔)，指定每次運作狀態檢查執行之間的時間間隔 (以毫秒為單位)。
- 對於 Path (路徑)，指定運作狀態檢查請求的目的地路徑。只有在運作狀態檢查通訊協定為 `http`或 時，才會使用此值`http2`。其他通訊協定會忽略此值。
- 對於 Timeout period (逾時期間)，指定等待收到運作狀態檢查回應的時間 (以秒為單位)。
- 對於 Unhealthy threshold (運作不良閾值)，指定在宣告接聽程式運作不良之前，必須達到的運作狀態檢查連續失敗次數。

e. （選用）啟用 TLS 終止 – 設定其他虛擬節點如何使用 TLS 與此虛擬節點通訊。

- 針對 模式，選取您想要在接聽程式上設定 TLS 的模式。
- 針對憑證方法，選取下列其中一個選項。憑證必須符合特定要求。如需詳細資訊，請參閱憑證需求。
  - AWS Certificate Manager 託管 – 選取現有的憑證。
  - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 使用 Secret Discovery Service 擷取的秘密名稱。
  - 本機檔案託管 – 在部署 Envoy 代理的檔案系統上指定憑證鏈檔案的路徑，以及私有金鑰。
- （選用）選取需要用戶端憑證和下列其中一個選項，以在用戶端提供憑證時啟用相互 TLS 身分驗證。若要進一步了解相互 TLS，請參閱 App Mesh Mutual TLS 身分驗證文件。
  - Envoy Secret Discovery Service (SDS) 託管 – 輸入 Envoy 將使用 Secret Discovery Service 擷取的秘密名稱。
  - 本機檔案託管 – 在部署 Envoy 的檔案系統上指定憑證鏈檔案的路徑。
- （選用）輸入主體別名。若要新增其他 SANs，請選取新增 SAN。SANs必須為 FQDN 或 URI 格式。

f. （選用）逾時

> **Note**
>
> 如果您指定的逾時大於預設值，請務必設定虛擬路由器和逾時大於預設值的路由。不過，如果您將逾時減少到低於預設值的值，您可以選擇在 Route 更新逾時。如需詳細資訊，請參閱路由。

- 請求逾時 – 如果您為接聽程式的通訊協定選取 grpc、http 或 http2，則可以指定請求逾時。預設值為 15 秒。值為 0 會停用逾時。
- 閒置持續時間 – 您可以指定任何接聽程式通訊協定的閒置持續時間。預設為 300 秒。

12. 選擇建立虛擬節點以完成。

AWS CLI

使用 建立虛擬節點 AWS CLI。

使用下列命令和輸入 JSON 檔案建立使用 DNS 進行服務探索的虛擬節點 （使用您自己的值取代*##值*）：

1.
```
aws appmesh create-virtual-node \
--cli-input-json file://create-virtual-node-dns.json
```

2. create-virtual-node-dns.json 範例的內容：

```
{
    "meshName": "meshName",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceBv1.svc.cluster.local"
            }
        }
    },
    "virtualNodeName": "nodeName"
}
```

3. 輸出範例：

```
{
```

```
    "virtualNode": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
            "createdAt": "2022-04-06T09:12:24.348000-05:00",
            "lastUpdatedAt": "2022-04-06T09:12:24.348000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ],
            "serviceDiscovery": {
                "dns": {
                    "hostname": "serviceBv1.svc.cluster.local"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualNodeName": "nodeName"
    }
}
```

如需使用 AWS CLI for App Mesh 建立虛擬節點的詳細資訊，請參閱 AWS CLI 參考中的 create-virtual-node 命令。

刪除虛擬節點

> **Note**
>
> 如果虛擬節點在任何路由中指定為目標，或在任何虛擬服務中指定為提供者，則無法刪除該虛擬節點。

AWS Management Console

使用 刪除虛擬節點 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您想要從中刪除虛擬節點的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual nodes (虛擬節點)。
4. 在虛擬節點表格中，選擇您要刪除的虛擬節點，然後選擇刪除。若要刪除虛擬節點，您的帳戶 ID 必須列在虛擬節點的網格擁有者或資源擁有者欄中。
5. 在確認方塊中，輸入 **`delete`**，然後選取刪除。

AWS CLI

使用 刪除虛擬節點 AWS CLI

1. 使用下列命令來刪除您的虛擬節點 （ 使用您自己的值取代*##*值 ）：

```
aws appmesh delete-virtual-node \
     --mesh-name meshName \
     --virtual-node-name nodeName
```

2. 輸出範例：

```
{
    "virtualNode": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
            "createdAt": "2022-04-06T09:12:24.348000-05:00",
            "lastUpdatedAt": "2022-04-07T11:03:48.120000-05:00",
```

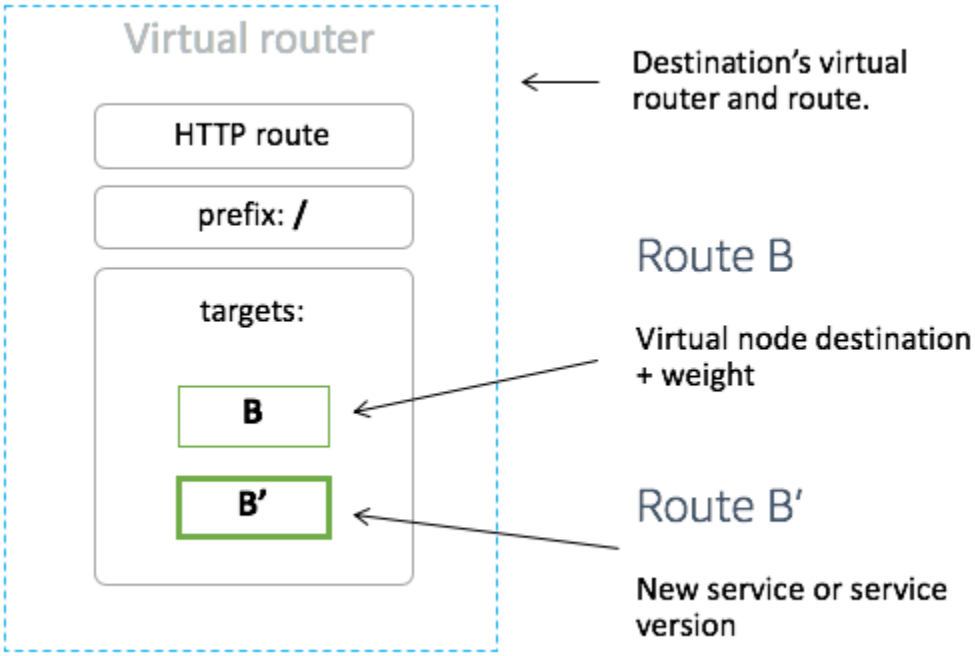
```
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "backends": [],
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ],
            "serviceDiscovery": {
                "dns": {
                    "hostname": "serviceBv1.svc.cluster.local"
                }
            }
        },
        "status": {
            "status": "DELETED"
        },
        "virtualNodeName": "nodeName"
    }
}
```

如需使用 AWS CLI for App Mesh 刪除虛擬節點的詳細資訊，請參閱 AWS CLI 參考中的 delete-virtual-node 命令。

虛擬路由器

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

虛擬路由器會處理網格內一或多個虛擬服務的流量。在您建立虛擬路由器後，您可以為虛擬路由器建立路由並將它們相關聯，它們會將傳入請求路由傳送到不同的虛擬節點。



虛擬路由器預期的任何傳入流量都應指定為接聽程式。

建立虛擬路由器

AWS Management Console

使用 建立虛擬路由器 AWS Management Console

> **Note**
>
> 建立 Virtual Router 時，您必須新增具有標籤的命名空間選擇器，以識別將 Routes 與建立的 Virtual Router 建立關聯的命名空間清單。

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要建立虛擬路由器的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual routers (虛擬路由器)。
4. 選擇 Create virtual router (建立虛擬路由器)。

5. 對於 Virtual router name (虛擬路由器名稱)，為您的虛擬路由器指定名稱。最多允許 255 個字母、數字、連字號和底線。
6. （選用）對於接聽程式組態，請為您的虛擬路由器指定連接埠和通訊協定。http 接聽程式允許將連線轉換為 Websocket。您可以按一下新增接聽程式來新增多個接聽程式。移除按鈕會移除該接聽程式。
7. 選擇 Create virtual router (建立虛擬路由器) 以完成。

AWS CLI

使用 建立虛擬路由器 AWS CLI。

使用下列命令建立虛擬路由器並輸入 JSON（使用您自己的值取代##值）：

1.
```
aws appmesh create-virtual-router \
     --cli-input-json file://create-virtual-router.json
```
2. create-virtual-router.json 範例的內容
3.
```
{
    "meshName": "meshName",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http"
                }
            }
        ]
    },
    "virtualRouterName": "routerName"
}
```
4. 輸出範例：

```
{
    "virtualRouter": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName",
```

```
            "createdAt": "2022-04-06T11:49:47.216000-05:00",
            "lastUpdatedAt": "2022-04-06T11:49:47.216000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "routerName"
    }
}
```


如需使用 AWS CLI for App Mesh 建立虛擬路由器的詳細資訊，請參閱 AWS CLI 參考中的 create-virtual-router 命令。

刪除虛擬路由器

> **Note**
>
> 如果虛擬路由器有任何路由，或指定為任何虛擬服務的提供者，則無法刪除該虛擬路由器。

AWS Management Console

使用 刪除虛擬路由器 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您想要從中刪除虛擬路由器的網格。會列出您擁有和已與您共用的所有網格。

3. 在左側導覽中，選擇 Virtual routers (虛擬路由器)。
4. 在虛擬路由器資料表中，選擇您要刪除的虛擬路由器，然後選取右上角的刪除。若要刪除虛擬路由器，您的帳戶 ID 必須列在虛擬路由器的網格擁有者或資源擁有者欄中。
5. 在確認方塊中，輸入 **delete**，然後按一下刪除。

AWS CLI

使用 刪除虛擬路由器 AWS CLI

1. 使用下列命令來刪除您的虛擬路由器 （ 使用您自己的值取代*##*值 ）：

```
aws appmesh delete-virtual-router \
     --mesh-name meshName \
     --virtual-router-name routerName
```

2. 輸出範例：

```
{
    "virtualRouter": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName",
            "createdAt": "2022-04-06T11:49:47.216000-05:00",
            "lastUpdatedAt": "2022-04-07T10:49:53.402000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
```

```
            "status": "DELETED"
        },
        "virtualRouterName": "routerName"
    }
}
```

如需使用 AWS CLI for App Mesh 刪除虛擬路由器的詳細資訊，請參閱 AWS CLI 參考中的 delete-virtual-router 命令。

路由

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

路由與虛擬路由器相關聯。此路由用於比對虛擬路由器的請求，並將流量分配到其相關聯的虛擬節點。如果路由符合請求，則可以將流量分配到一或多個目標虛擬節點。您可以為每個虛擬節點指定相對權重。本主題可協助您使用服務網格中的路由。

建立路由

AWS Management Console

使用 建立路由 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要建立路由的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual routers (虛擬路由器)。
4. 選擇您要與新路由建立關聯的虛擬路由器。如果沒有列出，則需要先建立虛擬路由器。
5. 在 Routes (路由) 表中，選擇 Create route (建立路由)。若要建立路由，您的帳戶 ID 必須列為路由的資源擁有者。
6. 對於 Route name (路由名稱)，指定您的路由要使用的名稱。
7. 針對路由類型，選擇您要路由的通訊協定。您選取的通訊協定必須符合您為虛擬路由器選取的接聽程式通訊協定，以及您路由流量的虛擬節點。

8. （選用）針對路由優先順序，指定要用於路由的優先順序，範圍為 0-1000。將依指定值比對路由，0 為最高優先順序。
9. （選用）選擇其他組態。從下面的通訊協定中，選擇您為 Route 類型選取的通訊協定，並視需要在主控台中指定設定。
10. 針對目標組態，選取現有的 App Mesh 虛擬節點以將流量路由到 ，並指定權重。您可以選擇新增目標以新增其他目標。所有目標的百分比最多必須加 100。如果未列出任何虛擬節點，則您必須先建立一個。如果選取的虛擬節點有多個接聽程式，則需要目標連接埠。
11. 針對相符組態，指定：

    比對組態不適用於 *tcp*

    - 如果 http/http2 是選取的類型：
      - （選用）方法 - 指定要在傳入 http/http2 請求中比對的方法標頭。
      - （選用）連接埠比對 - 比對傳入流量的連接埠。如果此虛擬路由器有多個接聽程式，則需要連接埠比對。
      - （選用）Prefix/Exact/Regex - 符合 URL 路徑的方法。
        - 字首比對 - 根據預設/，閘道路由的相符請求會重寫至目標虛擬服務的名稱，相符字首則會重寫至 。視您設定虛擬服務的方式而定，它可以使用虛擬路由器，根據特定字首或標頭，將請求路由到不同的虛擬節點。

          > **Note**
          >
          > 如果您啟用以路徑/字首為基礎的比對，App Mesh 會啟用路徑標準化 (normalize_path 和 merge_slashes)，以將路徑混淆漏洞的可能性降至最低。當參與請求的各方使用不同的路徑表示法時，會發生路徑混淆漏洞。

        - 完全相符 - 精確參數會停用路由的部分相符，並確保只有在路徑符合目前 URL 時，才會傳回路由。
        - Regex 比對 - 用於描述多個 URLs可能實際識別網站上的單一頁面的模式。
      - （選用）查詢參數 - 此欄位可讓您比對查詢參數。
      - （選用）標頭 - 指定 http 和 http2 的標頭。它應該符合傳入請求，以路由到目標虛擬服務。
    - 如果 grpc 是選取的類型：
      - 服務名稱 - 要比對請求的目的地服務。
      - 方法名稱 - 符合請求的目的地方法。

- （選用）中繼資料 - Match 根據中繼資料的存在來指定 。所有 必須符合才能處理請求。

12. 選取建立路由。

AWS CLI

使用 建立路由 AWS CLI。

使用下列命令和輸入 JSON 建立 gRPC 路由（使用您自己的值取代##值）：

1.
```
aws appmesh create-route \
     --cli-input-json file://create-route-grpc.json
```

2. create-route-grpc.json 範例的內容

```
{
    "meshName" : "meshName",
    "routeName" : "routeName",
    "spec" : {
       "grpcRoute" : {
          "action" : {
             "weightedTargets" : [
                {
                   "virtualNode" : "nodeName",
                   "weight" : 100
                }
             ]
          },
          "match" : {
             "metadata" : [
                {
                   "invert" : false,
                   "match" : {
                      "prefix" : "123"
                   },
                   "name" : "myMetadata"
                }
             ],
             "methodName" : "nameOfmethod",
             "serviceName" : "serviceA.svc.cluster.local"
          },
          "retryPolicy" : {
             "grpcRetryEvents" : [ "deadline-exceeded" ],
```

```
            "httpRetryEvents" : [ "server-error", "gateway-error" ],
            "maxRetries" : 3,
            "perRetryTimeout" : {
               "unit" : "s",
               "value" : 15
            },
            "tcpRetryEvents" : [ "connection-error" ]
         }
      },
      "priority" : 100
   },
   "virtualRouterName" : "routerName"
}
```

3. 輸出範例：

```
{
    "route": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName/route/routeName",
            "createdAt": "2022-04-06T13:48:20.749000-05:00",
            "lastUpdatedAt": "2022-04-06T13:48:20.749000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "routeName": "routeName",
        "spec": {
            "grpcRoute": {
                "action": {
                    "weightedTargets": [
                        {
                            "virtualNode": "nodeName",
                            "weight": 100
                        }
                    ]
                },
                "match": {
                    "metadata": [
                        {
                            "invert": false,
```

```
                            "match": {
                                "prefix": "123"
                            },
                            "name": "myMetadata"
                        }
                    ],
                    "methodName": "nameOfMehod",
                    "serviceName": "serviceA.svc.cluster.local"
                },
                "retryPolicy": {
"grpcRetryEvents": [
                        "deadline-exceeded"
                    ],
                    "httpRetryEvents": [
                        "server-error",
                        "gateway-error"
                    ],
                    "maxRetries": 3,
                    "perRetryTimeout": {
                        "unit": "s",
                        "value": 15
                    },
                    "tcpRetryEvents": [
                        "connection-error"
                    ]
                }
            },
            "priority": 100
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "routerName"
    }
}
```

如需使用 AWS CLI for App Mesh 建立路由的詳細資訊，請參閱 AWS CLI 參考中的 create-route 命令。

gRPC

（選用）相符

- （選用）輸入要符合 請求的目的地服務的服務名稱。如果您未指定名稱，則會比對對任何 服務的請求。
- （選用）輸入要比對請求之目的地方法的方法名稱。如果您未指定名稱，則會比對任何方法的請求。如果您指定方法名稱，則必須指定服務名稱。

（選用）中繼資料

選擇 Add metadata (新增中繼資料)。

- （選用）輸入您要根據其路由的中繼資料名稱，選取相符類型，然後輸入相符值。選取反轉將符合相反項目。例如，如果您指定 的中繼資料名稱`myMetadata`、Exact 的相符類型、 的相符值`123`，然後選取反轉，則路由會針對中繼資料名稱開頭為 以外的任何請求進行比對`123`。
- （選用）選取新增中繼資料以新增最多十個中繼資料項目。

（選用）重試政策

重試政策可讓用戶端保護自己免於間歇性的網路故障或間歇性的伺服器端故障。重試政策是選用的，但建議使用。重試逾時值會定義每次重試的逾時（包括初始嘗試）。如果您未定義重試政策，則 App Mesh 可能會為每個路由自動建立預設政策。如需詳細資訊，請參閱預設路由重試政策。

- 針對重試逾時，輸入逾時持續時間的單位數量。如果您選擇任何通訊協定重試事件，則需要值。
- 針對重試逾時單位，選取單位。如果您選擇任何通訊協定重試事件，則需要值。
- 針對重試次數上限，輸入請求失敗時的重試次數上限。如果您選擇任何通訊協定重試事件，則需要值。我們建議值至少為兩個。
- 選取一或多個 HTTP 重試事件。我們建議至少選取 stream-error 和 gateway-error。
- 選取 TCP 重試事件。
- 選取一或多個 gRPC 重試事件。我們建議至少選取已取消和無法使用。

（選用）逾時

- 預設值為 15 秒。如果您指定重試政策，則您在此指定的持續時間應一律大於或等於重試持續時間乘以您在重試政策中定義的重試次數上限，以便您的重試政策完成。如果您指定的持續時間大於 15

秒，則請確保為任何虛擬節點目標的接聽程式指定的逾時也大於 15 秒。如需詳細資訊，請參閱虛擬節點。

- 值為 0 會停用逾時。
- 路由可閒置的時間上限。

HTTP 和 HTTP/2

（選用）相符

- 指定路由應相符的字首。例如，如果您的虛擬服務名稱是 service-b.local，而您希望路由以配對請求和 service-b.local/metrics，則字首應該為 /metrics。指定 會/路由所有流量。
- （選用）選取方法。
- （選用）選取結構描述。僅適用於 HTTP2 路由。

（選用）標頭

- （選用）選取新增標頭。輸入您要路由的標頭名稱，選取相符類型，然後輸入相符值。選取反轉將符合相反項目。例如，如果您指定名為 的標頭clientRequestId，123並選取反轉，則路由會比對具有開頭為 以外的任何標頭的任何請求123。
- （選用）選取新增標頭。您最多可以新增十個標頭。

（選用）重試政策

重試政策可讓用戶端保護自己免於間歇性的網路故障或間歇性的伺服器端故障。重試政策是選用的，但建議使用。重試逾時值會定義每次重試的逾時（包括初始嘗試）。如果您未定義重試政策，則 App Mesh 可能會為每個路由自動建立預設政策。如需詳細資訊，請參閱預設路由重試政策。

- 針對重試逾時，輸入逾時持續時間的單位數量。如果您選擇任何通訊協定重試事件，則需要值。
- 針對重試逾時單位，選取單位。如果您選擇任何通訊協定重試事件，則需要值。
- 針對重試次數上限，輸入請求失敗時的重試次數上限。如果您選擇任何通訊協定重試事件，則需要值。我們建議值至少為兩個。
- 選取一或多個 HTTP 重試事件。我們建議至少選取 stream-error 和 gateway-error。
- 選取 TCP 重試事件。

（選用）逾時

- 請求逾時 – 預設值為 15 秒。如果您指定重試政策，則您在此指定的持續時間應一律大於或等於重試持續時間乘以您在重試政策中定義的重試次數上限，以便您的重試政策完成。
- 閒置持續時間 – 預設值為 300 秒。
- 值為 `0` 會停用逾時。

> **Note**
>
> 如果您指定大於預設值的逾時，請確定為所有虛擬節點參與者的接聽程式指定的逾時也大於預設值。不過，如果您將逾時減少到低於預設值的值，您可以選擇更新虛擬節點的逾時。如需詳細資訊，請參閱虛擬節點。

TCP

（選用）逾時

- 閒置持續時間 – 預設值為 300 秒。
- 值為 `0` 會停用逾時。

刪除路由

AWS Management Console

使用 刪除路由 AWS Management Console

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要從中刪除路由的網格。會列出您擁有和已與您共用的所有網格。
3. 在左側導覽中，選擇 Virtual routers (虛擬路由器)。
4. 選擇您要從中刪除路由的路由器。
5. 在路由表中，選擇您要刪除的路由，然後選取右上角的刪除。
6. 在確認方塊中，輸入 **`delete`**，然後按一下刪除。

AWS CLI

使用 刪除路由 AWS CLI

1. 使用以下命令刪除您的路由 （使用您自己的值取代*##*值）：

```
aws appmesh delete-route \
     --mesh-name meshName \
     --virtual-router-name routerName \
     --route-name routeName
```

2. 輸出範例：

```
{
    "route": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName/route/routeName",
            "createdAt": "2022-04-06T13:46:54.750000-05:00",
            "lastUpdatedAt": "2022-04-07T10:43:57.152000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "routeName": "routeName",
        "spec": {
            "grpcRoute": {
                "action": {
                    "weightedTargets": [
                        {
                            "virtualNode": "nodeName",
                            "weight": 100
                        }
                    ]
                },
                "match": {
                    "metadata": [
                        {
                            "invert": false,
                            "match": {
                                "prefix": "123"
```

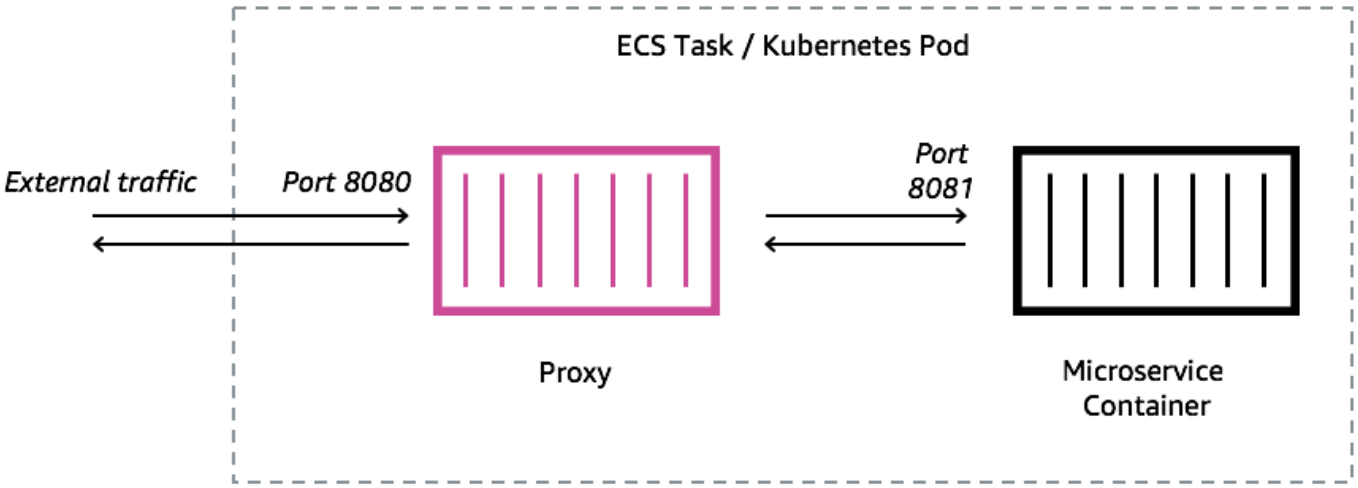
```
                            },
                            "name": "myMetadata"
                        }
                    ],
                    "methodName": "methodName",
                    "serviceName": "serviceA.svc.cluster.local"
                },
                "retryPolicy": {
                    "grpcRetryEvents": [
                        "deadline-exceeded"
                    ],
                    "httpRetryEvents": [
                        "server-error",
                        "gateway-error"
                    ],
                    "maxRetries": 3,
                    "perRetryTimeout": {
                        "unit": "s",
                        "value": 15
                    },
                    "tcpRetryEvents": [
                        "connection-error"
                    ]
                }
            },
            "priority": 100
        },
        "status": {
            "status": "DELETED"
        },
        "virtualRouterName": "routerName"
    }
}
```

如需使用 AWS CLI for App Mesh 刪除路由的詳細資訊，請參閱 AWS CLI 參考中的 delete-route 命令。

Envoy 影像

> **Important**
>
> 支援終止通知：在 2026 年 9 月 30 日， AWS 將停止對 的支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS App Mesh 是以 Envoy 代理為基礎的服務網格。



您必須將 Envoy 代理新增至 App Mesh 端點所代表的 Amazon ECS 任務、Kubernetes Pod 或 Amazon EC2 執行個體，例如虛擬節點或虛擬閘道。App Mesh 提供 Envoy 代理容器映像，並修補最新的漏洞和效能更新。App Mesh 會根據 App Mesh 功能集測試每個新的 Envoy 代理版本，然後再為您提供新的映像。

Envoy 映像變體

App Mesh 提供兩種 Envoy 代理容器映像變體。兩者之間的差異在於 Envoy 代理如何與 App Mesh 資料平面通訊，以及 Envoy 代理如何互相通訊。一個是與標準 App Mesh 服務端點通訊的標準映像。另一個變體符合 FIPS 規範，可與 App Mesh FIPS 服務端點通訊，並在 App Mesh 服務之間的 TLS 通訊中強制執行 FIPS 密碼編譯。

您可以從以下清單中選擇區域映像，或從我們名為 的公有儲存庫中選擇映像`aws-appmesh-envoy`。

Important

- 從 2023 年 6 月 30 日開始，只有 Envoy 映像`v1.17.2.0-prod`或更新版本與 App Mesh 相容。對於在 之前使用 Envoy 映像的目前客戶`v1.17.2.0`，雖然現有的 Envoy 將繼續相容，但我們強烈建議遷移至最新版本。
- 最佳實務是，強烈建議定期將 Envoy 版本升級至最新版本。只有最新的 Envoy 版本會使用最新的安全修補程式、功能版本和效能改進進行驗證。
- 版本 `1.17` 是 Envoy 的重大更新。如需詳細資訊，請參閱更新/遷移至 Envoy 1.17。
- 版本 `1.20.0.1`或更新版本ARM64相容。
- 如需IPv6支援，需要 Envoy 版本 `1.20` 或更新版本。

Note

FIPS 僅適用於美國和加拿大的 區域。

所有支援的區域都可以將`###`取代為 `me-south-1`、`ap-east-1`、`ap-southeast-3`、`eu-south-1`、 `il-central-1`和 以外的任何區域`af-south-1`。

標準

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

符合 FIPS 規範

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod-fips
```

`me-south-1`

標準

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

`ap-east-1`

標準

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

ap-southeast-3

標準

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

eu-south-1

標準

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

il-central-1

標準

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

af-south-1

標準

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

Public repository

標準

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.12.1-prod
```

符合 FIPS 規範

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.12.1-prod-fips
```

> **Note**
>
> 我們建議將 512 個 CPU 單位和至少 64 MiB 的記憶體配置給 Envoy 容器。在 Fargate 上，您可以設定的最低記憶體量為 1024 MiB 的記憶體。如果容器洞見或其他指標指出由於負載較高而導致資源不足，則可以增加 Envoy 容器的資源配置。

> **Note**
>
> 所有從 開始`aws-appmesh-envoy`的映像發行版本`v1.22.0.0`都會建置為無痕的 Docker 映像。我們進行這項變更，以便減少映像大小，並減少映像中未使用套件的漏洞暴露。如果您是在 aws-appmesh-envoy 映像的基礎上建置，並且依賴某些 AL2 基礎套件 （ 例如 yum) 和功能，則建議您從`aws-appmesh-envoy`映像內複製二進位檔，以使用 AL2 基礎建置新的 Docker 映像。
>
> 執行此指令碼以產生具有 標籤的自訂 Docker 映像 `aws-appmesh-envoy:v1.22.0.0-prod-al2`:

```
cat << EOF > Dockerfile
FROM public.ecr.aws/appmesh/aws-appmesh-envoy:v1.22.0.0-prod as envoy

FROM public.ecr.aws/amazonlinux/amazonlinux:2
RUN yum -y update && \
    yum clean all && \
    rm -rf /var/cache/yum

COPY --from=envoy /usr/bin/envoy /usr/bin/envoy
COPY --from=envoy /usr/bin/agent /usr/bin/agent
COPY --from=envoy /aws_appmesh_aggregate_stats.wasm /
aws_appmesh_aggregate_stats.wasm

CMD [ "/usr/bin/agent" ]
EOF

docker build -f Dockerfile -t aws-appmesh-envoy:v1.22.0.0-prod-al2 .
```

在 Amazon ECR 中存取此容器映像是由 AWS Identity and Access Management (IAM) 控制。因此，您必須使用 IAM 來驗證您是否具有 Amazon ECR 的讀取存取權。例如，使用 Amazon ECS 時，您可以將適當的任務執行角色指派給 Amazon ECS 任務。如果您使用限制存取特定 Amazon ECR 資源的

IAM 政策，請務必確認您允許存取識別aws-appmesh-envoy儲存庫的區域特定 Amazon Resource Name (ARN)。例如，在 us-west-2區域中，您可以允許存取下列資源：arn:aws:ecr:us-west-2:840364872350:repository/aws-appmesh-envoy。如需更多資訊，請參閱 Amazon ECR 受管政策。如果您在 Amazon EC2 執行個體上使用 Docker，請向儲存庫驗證 Docker。如需詳細資訊，請參閱登錄檔身分驗證。

我們偶爾會發佈新的 App Mesh 功能，這些功能取決於尚未合併到上游 Envoy 映像的 Envoy 變更。若要在 Envoy 變更在上游合併之前使用這些新的 App Mesh 功能，您必須使用 App Mesh 提供的 Envoy 容器映像。如需變更清單，請參閱 Envoy Upstream標籤的 App Mesh GitHub 藍圖問題。我們建議您使用 App Mesh Envoy 容器映像作為最佳支援選項。

Envoy 組態變數

Important

支援終止通知：在 2026 年 9 月 30 日， AWS 將停止對 的支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

使用下列環境變數來設定 App Mesh 虛擬節點任務群組的 Envoy 容器。

Note

App Mesh Envoy 1.17 不支援 Envoy 的 v2 xDS API。如果您使用的是接受 Envoy 組態檔案的 Envoy 組態變數，則必須將其更新為最新的 v3 xDS API。

必要變數

所有 App Mesh Envoy 容器都需要下列環境變數。此變數只能與版本 1.15.0或更新版本的 Envoy 映像搭配使用。如果您使用的是舊版映像，則必須改為設定 APPMESH_VIRTUAL_NODE_NAME變數。

APPMESH_RESOURCE_ARN

當您將 Envoy 容器新增至任務群組時，請將此環境變數設定為虛擬節點的 ARN 或任務群組代表的虛擬閘道。下列清單包含範例 ARNs：

- 虛擬節點 – arn：aws：appmesh：*Region-code*：*111122223333*：mesh/*meshName*/virtualNode/*virtualNodeName*
- 虛擬閘道 – arn：aws：appmesh：*Region-code*：*111122223333*：mesh/*meshName*/virtualGateway/*virtualGatewayName*

選用變數

下列環境變數對於 App Mesh Envoy 容器是選用的。

`ENVOY_LOG_LEVEL`

指定 Envoy 容器的日誌層級。

有效值：`trace`、`debug`、`info`、 `warn`、`error`、`critical`、`off`

預設：`info`

`ENVOY_INITIAL_FETCH_TIMEOUT`

指定 Envoy 在初始化程序期間從管理伺服器等待第一個組態回應的時間量。

如需詳細資訊，請參閱 Envoy 文件中的組態來源。設為 時`0`，沒有逾時。

預設：`0`

`ENVOY_CONCURRENCY`

啟動 Envoy 時設定`--concurrency`命令列選項。預設不會設定此值。此選項可從 Envoy 版本 `v1.24.0.0-prod` 或更新版本取得。

如需詳細資訊，請參閱 Envoy 文件中的命令列選項。

管理員變數

使用這些環境變數來設定 Envoy 的管理界面。

`ENVOY_ADMIN_ACCESS_PORT`

指定 Envoy 要接聽的自訂管理員連接埠。預設：`9901`。

> **Note**
>
> Envoy 管理員連接埠應與虛擬閘道或虛擬節點上的任何接聽程式連接埠不同

`ENVOY_ADMIN_ACCESS_LOG_FILE`

指定要寫入 Envoy 存取日誌的自訂路徑。預設：`/tmp/envoy_admin_access.log`。

`ENVOY_ADMIN_ACCESS_ENABLE_IPV6`

切換 Envoy 的管理界面以接受IPv6流量，這可讓此界面同時接受 IPv4和 IPv6流量。根據預設，此旗標設定為 false，Envoy 只會接聽IPv4流量。此變數只能與 Envoy 映像版本 1.22.0 或更新版本搭配使用。

代理程式變數

使用這些環境變數來設定 AWS App Mesh Agent for Envoy。如需詳細資訊，請參閱 App Mesh Agent for Envoy。

`APPNET_ENVOY_RESTART_COUNT`

指定代理程式在執行中的任務或 Pod 中重新啟動 Envoy 代理程序的次數，如果結束的話。客服人員也會在每次 Envoy 結束時記錄結束狀態，以簡化故障診斷。此變數的預設值為 0。設定預設值時，代理程式不會嘗試重新啟動程序。

預設：`0`

上限：`10`

`PID_POLL_INTERVAL_MS`

指定代理程式檢查 Envoy 代理程序狀態的間隔，以毫秒為單位。預設值為 100。

預設：`100`

下限：`100`

上限：`1000`

`LISTENER_DRAIN_WAIT_TIME_S`

指定 Envoy 代理在程序結束之前等待作用中連線關閉的時間，以秒為單位。

預設：20

下限：5

上限：110

APPNET_AGENT_ADMIN_MODE

啟動客服人員的管理界面伺服器，並將其繫結至 tcp 地址或 unix 通訊端。

有效值：tcp、uds

APPNET_AGENT_HTTP_PORT

在 tcp 模式中指定用於繫結客服人員管理界面的連接埠。1024 如果 ！= uid ，請確保連接埠值為 >0。確保連接埠小於 65535。

預設：9902

APPNET_AGENT_ADMIN_UDS_PATH

在 uds 模式下為客服人員的管理界面指定 unix 網域通訊端路徑。

預設：/var/run/ecs/appnet_admin.sock

追蹤變數

您可以設定無或下列任一追蹤驅動程式。

AWS X-Ray 變數

使用下列環境變數來設定 App Mesh AWS X-Ray。如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》。

ENABLE_ENVOY_XRAY_TRACING

使用 127.0.0.1:2000做為預設協助程式端點來啟用 X-Ray 追蹤。若要啟用，請將 值設定為 1。預設值為 0。

XRAY_DAEMON_PORT

指定連接埠值以覆寫預設的 X-Ray 協助程式連接埠：2000。

`XRAY_SAMPLING_RATE`

指定取樣率以覆寫 X-Ray 追蹤器的預設取樣率 `0.05`(5%)。將值指定為介於 `0`和 `1.00`(100%) 之間的小數。如果`XRAY_SAMPLING_RULE_MANIFEST`指定 ，則會覆寫此值。版本 `v1.19.1.1-prod`和更新版本的 Envoy 映像支援此變數。

`XRAY_SAMPLING_RULE_MANIFEST`

在 Envoy 容器檔案系統中指定檔案路徑，以設定 X-Ray 追蹤器的當地語系化自訂取樣規則。如需詳細資訊，請參閱《 AWS X-Ray 開發人員指南》中的取樣規則。版本 `v1.19.1.0-prod`和更新版本的 Envoy 映像支援此變數。

`XRAY_SEGMENT_NAME`

指定追蹤的區段名稱，以覆寫預設的 X-Ray 區段名稱。根據預設，此值會設定為 `mesh/resourceName`。Envoy 映像版本 `v1.23.1.0-prod`或更新版本支援此變數。

Datadog 追蹤變數

下列環境變數可協助您使用 Datadog 代理程式追蹤器設定 App Mesh。如需詳細資訊，請參閱 Datadog 文件中的客服人員組態。

`ENABLE_ENVOY_DATADOG_TRACING`

啟用使用 `127.0.0.1:8126`做為預設 Datadog 代理程式端點的 Datadog 追蹤收集。若要啟用，請將值設定為 `1` (預設值為 `0`)。

`DATADOG_TRACER_PORT`

指定連接埠值以覆寫預設 Datadog 代理程式連接埠：`8126`。

`DATADOG_TRACER_ADDRESS`

指定 IP 地址以覆寫預設 Datadog 代理程式地址：`127.0.0.1`。

`DD_SERVICE`

指定追蹤的服務名稱，以覆寫預設的 Datadog 服務名稱：`envoy-meshName/virtualNodeName`。版本 `v1.18.3.0-prod`和更新版本的 Envoy 映像支援此變數。

Jaeger 追蹤變數

使用下列環境變數來設定 App Mesh 搭配 Jaeger 追蹤。如需詳細資訊，請參閱 Jaeger 文件中的入門。版本 `1.16.1.0-prod`和更新版本的 Envoy 映像支援這些變數。

`ENABLE_ENVOY_JAEGER_TRACING`

啟用使用 `127.0.0.1:9411`做為預設 Jaeger 端點的 Jaeger 追蹤集合。若要啟用，請將值設定為 `1`（預設值為 `0`)。

`JAEGER_TRACER_PORT`

指定連接埠值以覆寫預設的 Jaeger 連接埠：`9411`。

`JAEGER_TRACER_ADDRESS`

指定 IP 地址以覆寫預設的 Jaeger 地址：`127.0.0.1`。

`JAEGER_TRACER_VERSION`

指定收集器是否需要 `JSON`或`PROTO`編碼格式的追蹤。根據預設，此值會設為 `PROTO`。Envoy 映像版本 `v1.23.1.0-prod`或更新版本支援此變數。

Envoy 追蹤變數

設定下列環境變數以使用您自己的追蹤組態。

`ENVOY_TRACING_CFG_FILE`

在 Envoy 容器檔案系統中指定檔案路徑。如需詳細資訊，請參閱 Envoy 文件`config.trace.v3.Tracing`中的 。

> **Note**
>
> 如果追蹤組態需要指定追蹤叢集，請務必在相同的追蹤組態檔案中，於 `static_resources`下設定相關聯的叢集組態。例如，Zipkin 有一個託管追蹤收集器的叢集名稱`collector_cluster`欄位，該叢集需要靜態定義。

DogStatsD 變數

使用下列環境變數來設定 App Mesh 搭配 DogStatsD。如需詳細資訊，請參閱 DogStatsD 文件。

`ENABLE_ENVOY_DOG_STATSD`

使用 `127.0.0.1:8125`做為預設協助程式端點來啟用 DogStatsD 統計資料。若要啟用，請將 值設定為 `1`。

`STATSD_PORT`

指定連接埠值以覆寫預設的 DogStatsD 協助程式連接埠。

`STATSD_ADDRESS`

指定 IP 地址值以覆寫預設的 DogStatsD 協助程式 IP 地址。預設：`127.0.0.1`。此變數只能與版本 `1.15.0`或更新版本的 Envoy 映像搭配使用。

`STATSD_SOCKET_PATH`

指定 DogStatsD 協助程式的 unix 網域通訊端。如果未指定此變數並啟用 DogStatsD，則此值預設為 的 DogStatsD 協助程式 IP 地址連接埠`127.0.0.1:8125`。如果`ENVOY_STATS_SINKS_CFG_FILE`指定變數包含統計資料接收器組態，則會覆寫所有 DogStatsD 變數。Envoy 映像版本 `v1.19.1.0-prod`或更新版本支援此變數。

App Mesh 變數

下列變數可協助您設定 App Mesh。

`APPMESH_RESOURCE_CLUSTER`

根據預設，App Mesh 會使用您在 Envoy 在指標和追蹤中參考本身`APPMESH_RESOURCE_ARN`時在中指定的資源名稱。您可以藉由使用自己的名稱設定 `APPMESH_RESOURCE_CLUSTER` 環境變數，以覆寫此行為。此變數只能與版本 `1.15.0`或更新版本的 Envoy 映像搭配使用。

`APPMESH_METRIC_EXTENSION_VERSION`

將 值設定為 `1`以啟用 App Mesh 指標延伸。如需使用 App Mesh 指標延伸的詳細資訊，請參閱 App Mesh 的指標延伸。

`APPMESH_DUALSTACK_ENDPOINT`

將 值設定為 `1` 以連線至 App Mesh Dual Stack 端點。設定此旗標時，Envoy 會使用支援雙堆疊的網域。根據預設，此旗標設定為 false，且只會連線到我們的`IPv4`網域。此變數只能與 Envoy 映像版本 1.22.0 或更新版本搭配使用。

Envoy 統計資料變數

使用下列環境變數來設定 App Mesh 搭配 Envoy Stats。如需詳細資訊，請參閱 Envoy Stats 文件。

`ENABLE_ENVOY_STATS_TAGS`

啟用 App Mesh 定義的標籤 `appmesh.mesh`和 的使用`appmesh.virtual_node`。如需詳細資訊，請參閱 Envoy 文件中的 config.metrics.v3.TagSpecifier。若要啟用，請將 值設定為 1。

`ENVOY_STATS_CONFIG_FILE`

在 Envoy 容器檔案系統中指定檔案路徑，以使用您自己的檔案覆寫預設 Stats 標籤組態檔案。如需詳細資訊，請參閱 config.metrics.v3.StatsConfig。

> **Note**
>
> 設定包含統計資料篩選條件的自訂統計資料組態，可能會導致 Envoy 進入不再與世界 App Mesh 狀態正確同步的狀態。這是 Envoy 中的錯誤。我們建議不要在 Envoy 中執行任何統計資料篩選。如果篩選是絕對必要的，我們在藍圖上列出了此問題中的幾個解決方法。

`ENVOY_STATS_SINKS_CFG_FILE`

在 Envoy 容器檔案系統中指定檔案路徑，以使用您自己的組態覆寫預設組態。如需詳細資訊，請參閱 Envoy 文件中的 config.metrics.v3.StatsSink。

已棄用變數

Envoy 版本 `APPMESH_RESOURCE_NAME` `1.15.0`或更新版本不再支援環境變數 `APPMESH_VIRTUAL_NODE_NAME` 和 。不過，現有網格仍支援這些網格。將 `APPMESH_RESOURCE_ARN`用於所有 App Mesh 端點，而不是將這些變數與 Envoy 版本 `1.15.0` 或更新版本搭配使用。

App Mesh 設定的 Envoy 預設值

> **Important**
>
> 支援終止通知：在 2026 年 9 月 30 日， AWS 將停止對 的支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

下列各節提供 App Mesh 所設定之路由重試政策和斷路器的 Envoy 預設值的相關資訊。

預設路由重試政策

如果您在 2020 年 7 月 29 日之前帳戶中沒有網格，App Mesh 會在 2020 年 7 月 29 日當天或之後，自動為帳戶中任何網格中的所有 HTTP、HTTP/2 和 gRPC 請求建立預設 Envoy 路由重試政策。如果您在 2020 年 7 月 29 日之前在帳戶中有任何網格，則不會為 2020 年 7 月 29 日以前、當天或之後存在的任何 Envoy 路由建立預設政策。除非您在 AWS 支援下開立票證，否則即為如此。支援處理票證後，會為 App Mesh 在處理票證當天或之後建立的任何未來 Envoy 路由建立預設政策。如需 Envoy 路由重試政策的詳細資訊，請參閱 Envoy 文件中的 config.route.v3.RetryPolicy。

當您建立 App Mesh 路由或定義 App Mesh 虛擬服務的虛擬節點提供者時，App Mesh 會建立 Envoy 路由。雖然您可以建立 App Mesh 路由重試政策，但您無法為虛擬節點提供者建立 App Mesh 重試政策。

預設政策無法透過 App Mesh API 顯示。預設政策只能透過 Envoy 顯示。若要檢視組態，請啟用 管理介面，並將 的請求傳送至 Envoyconfig_dump。預設政策包含下列設定：

- 重試次數上限 – 2
- gRPC 重試事件 – `UNAVAILABLE`
- HTTP 重試事件 – `503`

> **Note**
>
> 無法建立尋找特定 HTTP 錯誤碼的 App Mesh 路由重試政策。不過，App Mesh 路由重試政策可以尋找 `server-error`或 `gateway-error`。這兩者都包含`503`錯誤。如需詳細資訊，請參閱路由。

- TCP 重試事件 – `connect-failure`和 `refused-stream`

> **Note**
>
> 無法建立尋找其中一個事件的 App Mesh 路由重試政策。不過，App Mesh 路由重試政策可以尋找 `connection-error`，這相當於 `connect-failure`。如需詳細資訊，請參閱路由。

- 重設 – 如果上游伺服器完全沒有回應 (disconnect/reset/read逾時），Envoy 會嘗試重試。

預設斷路器

當您在 App Mesh 中部署 Envoy 時，Envoy 預設值會設定為某些斷路器設定。如需詳細資訊，請參閱 Envoy 文件中的 cluster.CircuitBreakers.Thresholds。這些設定無法透過 App Mesh API 顯示。這些設定只能透過 Envoy 顯示。若要檢視組態，請啟用管理介面，並將 的請求傳送至 Envoyconfig_dump。

如果在 2020 年 7 月 29 日之前您的帳戶中沒有網格，則對於您在 2020 年 7 月 29 日當天或之後建立的網格中部署的每個 Envoy，App Mesh 會透過變更後續設定的 Envoy 預設值來有效停用斷路器。如果您在 2020 年 7 月 29 日之前在帳戶中有任何網格，除非您在 AWS 支援下開立票證，否則將為您在 2020 年 7 月 29 日當天或之後部署的任何 Envoy 設定 Envoy 預設值。一旦支援處理票證，以下 Envoy 設定的 App Mesh 預設值將由 App Mesh 設定在您處理票證日期之後部署的所有 Envoy 上：

- **max_requests** – 2147483647
- **max_pending_requests** – 2147483647
- **max_connections** – 2147483647
- **max_retries** – 2147483647

> **Note**
>
> 無論您的 Envoy 具有 Envoy 或 App Mesh 預設斷路器值，您都無法修改這些值。

更新/遷移至 Envoy 1.17

> **Important**
>
> 支援終止通知：在 2026 年 9 月 30 日， AWS 將停止對 的支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

搭配 SPIRE 的 Secret Discovery Service

如果您使用 SPIRE (SPIFFE 執行期環境） 搭配 App Mesh 將信任憑證分發到服務，請確認您使用的是 0.12.0 SPIRE 代理程式的至少版本 (2020 年 12 月發行）。這是第一個可在 之後支援 Envoy 版本的版本1.16。

規則表達式變更

從 Envoy 開始1.17，App Mesh 預設會將 Envoy 設定為使用 RE2 規則表達式引擎。此變更對大多數使用者來說很明顯，但 Routes 或 Gateway Routes 中的相符項目不再允許規則表達式中的前瞻或後退參考。

正面和負面前瞻

正 - 正前向是以 開頭的括號表達式?=：

```
(?=example)
```

這些值在執行字串取代時具有最多的公用程式，因為它們允許比對字串，而不會在比對過程中耗用字元。由於 App Mesh 不支援 regex 字串取代，我們建議您將它們取代為一般相符項目。

```
(example)
```

負 - 負前向是以 開頭的括號表達式?!。

```
ex(?!amp)le
```

括號表達式用於宣告表達式的該部分與指定的輸入不相符。在大多數情況下，您可以將它們取代為零量化器。

```
ex(amp){0}le
```

如果表達式本身是字元類別，您可以否定整個類別，並使用 將其標記為選用?。

```
prefix(?![0-9])suffix => prefix[^0-9]?suffix
```

根據您的使用案例，您也可以變更路由以處理此問題。

```
{
    "routeSpec": {
        "priority": 0,
        "httpRoute": {
            "match": {
                "headers": [
                    {
```

```
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix(?!suffix)"
                        }
                    }
                ]
            }
        }
    }
}

{
    "routeSpec": {
        "priority": 1,
        "httpRoute": {
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix"
                        }
                    }
                ]
            }
        }
    }
}
```

第一個路由比對會尋找開頭為「字首」但後面沒有「尾碼」的標頭。第二個路由的作用是比對所有其他以「字首」開頭的標頭，包括以「尾碼」結尾的標頭。反之，這些也可以反轉為移除負面前瞻的方式。

```
{
    "routeSpec": {
        "priority": 0,
        "httpRoute": {
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix.*?suffix"
                        }
```

```
                    }
                ]
            }
        }
    }
}

{
    "routeSpec": {
        "priority": 1,
        "httpRoute": {
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix"
                        }
                    }
                ]
            }
        }
    }
}
```

此範例會反轉路由，為結尾為「字尾」的標頭提供更高的優先順序，而開頭為「字首」的所有其他標頭會在較低優先順序的路由中比對。

後端參考

回溯參考是透過重複先前括號的群組來撰寫較短表達式的一種方式。他們有此表單。

```
(group1)(group2)\1
```

\ 後面加上數字的反斜線做為表達式中第 n 個括號群組的預留位置。在此範例中， \1 會用作(group1)第二次寫入的替代方法。

```
(group1)(group2)(group1)
```

只需將反參考取代為範例中參考的群組，即可移除這些項目。

Envoy 的代理程式

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

代理程式是 Envoy 映像中的程序管理員，適用於 App Mesh。代理程式可確保 Envoy 保持執行狀態、保持運作狀態，並減少停機時間。它會篩選 Envoy 統計資料和輔助資料，以提供 Envoy 代理在 App Mesh 中操作的分割檢視。這可協助您更快速地對相關錯誤進行故障診斷。

您可以使用 代理程式來設定您希望在代理狀態不佳時重新啟動 Envoy 代理的次數。如果失敗，客服人員會在 Envoy 結束時記錄最終結束狀態。您可以在故障疑難排解時使用此功能。代理程式也促進 Envoy 連線耗盡，這有助於讓您的應用程式對故障更具彈性。

使用以下變數設定 Agent for Envoy：

- `APPNET_ENVOY_RESTART_COUNT` – 當此變數設定為非零值時，代理程式會嘗試重新啟動 Envoy 代理程序，直到其認為代理程序在輪詢狀態不佳時所設定的數字為止。這有助於在代理運作狀態檢查失敗時，相較於容器協調器的任務或 Pod 替換，提供更快的重新啟動速度，進而減少停機時間。
- `PID_POLL_INTERVAL_MS` – 設定此變數時，預設值會保留為 `100`。設定為此值時，相較於透過容器協調器運作狀態檢查的任務或 Pod 替換，您可以允許在 Envoy 程序結束時更快地偵測和重新啟動。
- `LISTENER_DRAIN_WAIT_TIME_S` – 設定此變數時，請考慮為停止任務或 Pod 而設定的容器協調程式逾時。例如，如果此值大於協調器逾時，Envoy 代理只能耗盡一段時間，直到協調器強制停止任務或 Pod。
- `APPNET_AGENT_ADMIN_MODE` – 當此變數設定為 `tcp`或 時`uds`，客服人員會提供本機管理介面。此管理界面可做為與 Envoy 代理互動的安全端點，並提供下列 APIs以進行運作狀態檢查、遙測資料，並摘要說明代理的操作條件。
  - `GET /status` – 查詢 Envoy 統計資料並傳回伺服器資訊。
  - `POST /drain_listeners` – 耗盡所有傳入接聽程式。
  - `POST /enableLogging?level=<desired_level>` – 變更所有記錄器的 Envoy 記錄層級。
  - `GET /stats/prometheus` – 以 Prometheus 格式顯示 Envoy 統計資料。
  - `GET /stats/prometheus?usedonly` – 僅顯示 Envoy 已更新的統計資料。

如需客服人員組態變數的詳細資訊，請參閱 Envoy 組態變數。

新的 AWS App Mesh 代理程式包含在從 版本開始的 App Mesh 最佳化 Envoy 映像中`1.21.0.0`，且不需要在客戶任務或 Pod 中配置其他資源。

App Mesh 可觀測性

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

使用 App Mesh 的好處之一是更清楚地了解您的微服務應用程式。App Mesh 能夠使用許多不同的記錄、指標和追蹤解決方案。

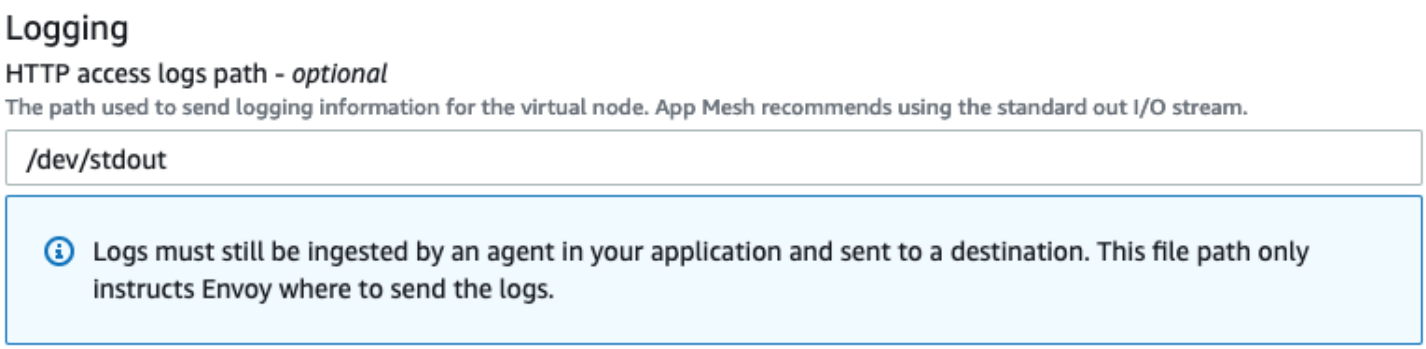
Envoy 代理和 App Mesh 提供下列類型的工具，可協助您更清楚地檢視應用程式和代理：

- 日誌
- 指標
- 追蹤

日誌

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

當您建立虛擬節點和虛擬閘道時，您可以選擇設定 Envoy 存取日誌。在 主控台中，這是在虛擬節點和虛擬閘道的記錄區段中建立或編輯工作流程。



上圖顯示適用於 Envoy 存取日誌/dev/stdout的 記錄路徑。

對於 format，請指定兩種可能格式之一，json或 text和 模式。 會json先取得金鑰對並將其轉換為 JSON 結構，再將其傳遞給 Envoy。

下列程式碼區塊顯示您可以在 中使用的 JSON 表示法 AWS CLI。

```
      "logging": {
         "accessLog": {
            "file": {
               "path": "/dev/stdout",
                "format" : {
                    // Exactly one of json or text should be specified
                    "json": [ // json will be implemented with key pairs
                        {
                            "key": "string",
                            "value": "string"
                        }
                    ]
                    "text": "string" //e.g. "%LOCAL_REPLY_BODY%:%RESPONSE_CODE%:path=
%REQ(:path)%\n"
                }
            }
         }
      }
```

Important

請務必檢查輸入模式是否適用於 Envoy，否則 Envoy 會拒絕更新，並將最新的變更存放在中error state。

當您將 Envoy 存取日誌傳送至 時/dev/stdout，它們會與 Envoy 容器日誌混合。您可以使用 等標準 Docker 日誌驅動程式，將它們匯出至日誌儲存和處理服務，例如 CloudWatch Logsawslogs。如需詳細資訊，請參閱《Amazon ECS 開發人員指南》中的使用 awslogs Log Driver。若要僅匯出 Envoy 存取日誌（並忽略其他 Envoy 容器日誌），您可以將 ENVOY_LOG_LEVEL 設定為 off。您可以包含格式字串 來記錄沒有查詢字串的請求%REQ_WITHOUT_QUERY(X?Y):Z%。如需範例，請參閱 ReqWithoutQuery Formatter。如需詳細資訊，請參閱 Envoy 文件中的存取日誌。

在 Kubernetes 上啟用存取日誌

使用適用於 Kubernetes 的 App Mesh Controller 時，您可以將記錄組態新增至虛擬節點規格，以使用存取記錄來設定虛擬節點，如下列範例所示。

```
---
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: virtual-node-name
  namespace: namespace
spec:
  listeners:
    - portMapping:
        port: 9080
        protocol: http
  serviceDiscovery:
    dns:
      hostName: hostname
  logging:
    accessLog:
      file:
        path: "/dev/stdout"
```

您的叢集必須有日誌轉送器才能收集這些日誌，例如 Fluentd。如需詳細資訊，請參閱將 Fluentd 設定為 DaemonSet，以將日誌傳送至 CloudWatch Logs。

Envoy 也會將各種偵錯日誌從其篩選條件寫入 stdout。這些日誌有助於深入了解 Envoy 與 App Mesh 的通訊，以及service-to-service流量。您可以使用 ENVOY_LOG_LEVEL環境變數來設定特定記錄層級。例如，以下文字來自範例偵錯日誌，其中顯示 Envoy 針對特定 HTTP 請求比對的叢集。

```
[debug][router] [source/common/router/router.cc:434] [C4][S17419808847192030829]
 cluster 'cds_ingress_howto-http2-mesh_color_client_http_8080' match for URL '/ping'
```

Firelens 和 Cloudwatch

Firelens 是容器日誌路由器，可用來收集 Amazon ECS 和 的日誌 AWS Fargate。您可以在我們的AWS 範例儲存庫中找到使用 Firelens 的範例。

您可以使用 CloudWatch 來收集記錄資訊和指標。您可以在 App Mesh 文件的匯出指標區段中找到 CloudWatch 的詳細資訊。

使用 Envoy 指標監控您的應用程式

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

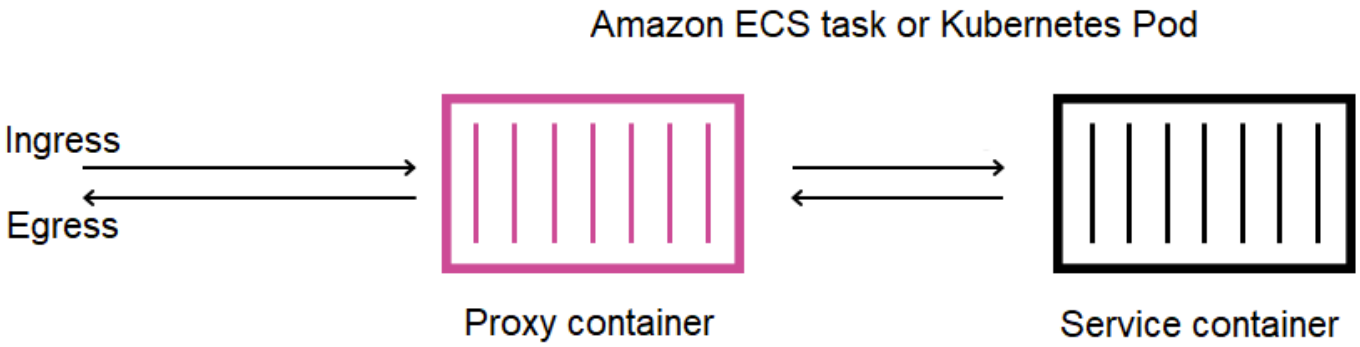
Envoy 將其指標分類為下列主要類別：

- 下游 - 與傳入代理的連線和請求相關的指標。
- 上游：與代理發出的傳出連線和請求相關的指標。
- Server - 描述 Envoy 內部狀態的指標。這些包括運作時間或配置記憶體等指標。

在 App Mesh 中，代理會攔截上游和下游流量。例如，從用戶端收到的請求，以及服務容器提出的請求，都被 Envoy 歸類為下游流量。為了區分這些不同類型的上游和下游流量，App Mesh 會根據相對於服務的流量方向進一步分類 Envoy 指標：

- 輸入 - 與流向服務容器的連線和請求相關的指標和資源。
- 輸出 - 與從服務容器流出，最終從您的 Amazon ECS 任務或 Kubernetes Pod 流出之連線和請求相關的指標和資源。

下圖顯示代理和服務容器之間的通訊。



資源命名慣例

了解 Envoy 如何檢視您的網格及其資源如何對應回您在 App Mesh 中定義的資源非常有用。以下是 App Mesh 設定的主要 Envoy 資源：

- 接聽程式 — 代理接聽下游連線的地址和連接埠。在上圖中，App Mesh 會為傳入 Amazon ECS 任務或 Kubernetes Pod 的流量建立傳入接聽程式，並為離開服務容器的流量建立輸出接聽程式。
- 叢集 - 代理連線和路由流量的上游端點具名群組。在 App Mesh 中，您的服務容器會呈現為叢集，以及服務可以連線的所有其他虛擬節點。
- Routes - 這些對應至您在網格中定義的路由。其中包含代理符合請求的條件，以及傳送請求的目標叢集。
- 端點和叢集負載指派 - 上游叢集的 IP 地址。使用 AWS Cloud Map 做為虛擬節點的服務探索機制時，App Mesh 會將探索的服務執行個體做為端點資源傳送到您的代理。
- 秘密 - 這些包括但不限於您的加密金鑰和 TLS 憑證。使用 AWS Certificate Manager 做為用戶端和伺服器憑證的來源時，App Mesh 會將公有和私有憑證做為秘密資源傳送至您的代理。

App Mesh 使用一致的機制來命名 Envoy 資源，您可以使用這些資源來與網格建立關聯。

了解接聽程式和叢集的命名機制對於了解 App Mesh 中的 Envoy 指標非常重要。

接聽程式名稱

接聽程式會使用下列格式命名：

```
lds_<traffic direction>_<listener IP address>_<listening port>
```

您通常會看到下列接聽程式在 Envoy 中設定：

- lds_ingress_0.0.0.0_15000
- lds_egress_0.0.0.0_15001

使用 Kubernetes CNI 外掛程式或 IP 資料表規則，Amazon ECS 任務或 Kubernetes Pod 中的流量會導向連接埠 15000和 15001。App Mesh 使用這兩個接聽程式設定 Envoy，以接受傳入（傳入）和傳出（傳出）流量。如果您在虛擬節點上未設定接聽程式，則不應看到傳入接聽程式。

叢集名稱

大多數叢集使用以下格式：

```
cds_<traffic direction>_<mesh name>_<virtual node name>_<protocol>_<port>
```

您的服務與每個 通訊的虛擬節點都有自己的叢集。如前所述，App Mesh 會為 Envoy 旁執行的服務建立叢集，讓代理程式可以傳送輸入流量給它。

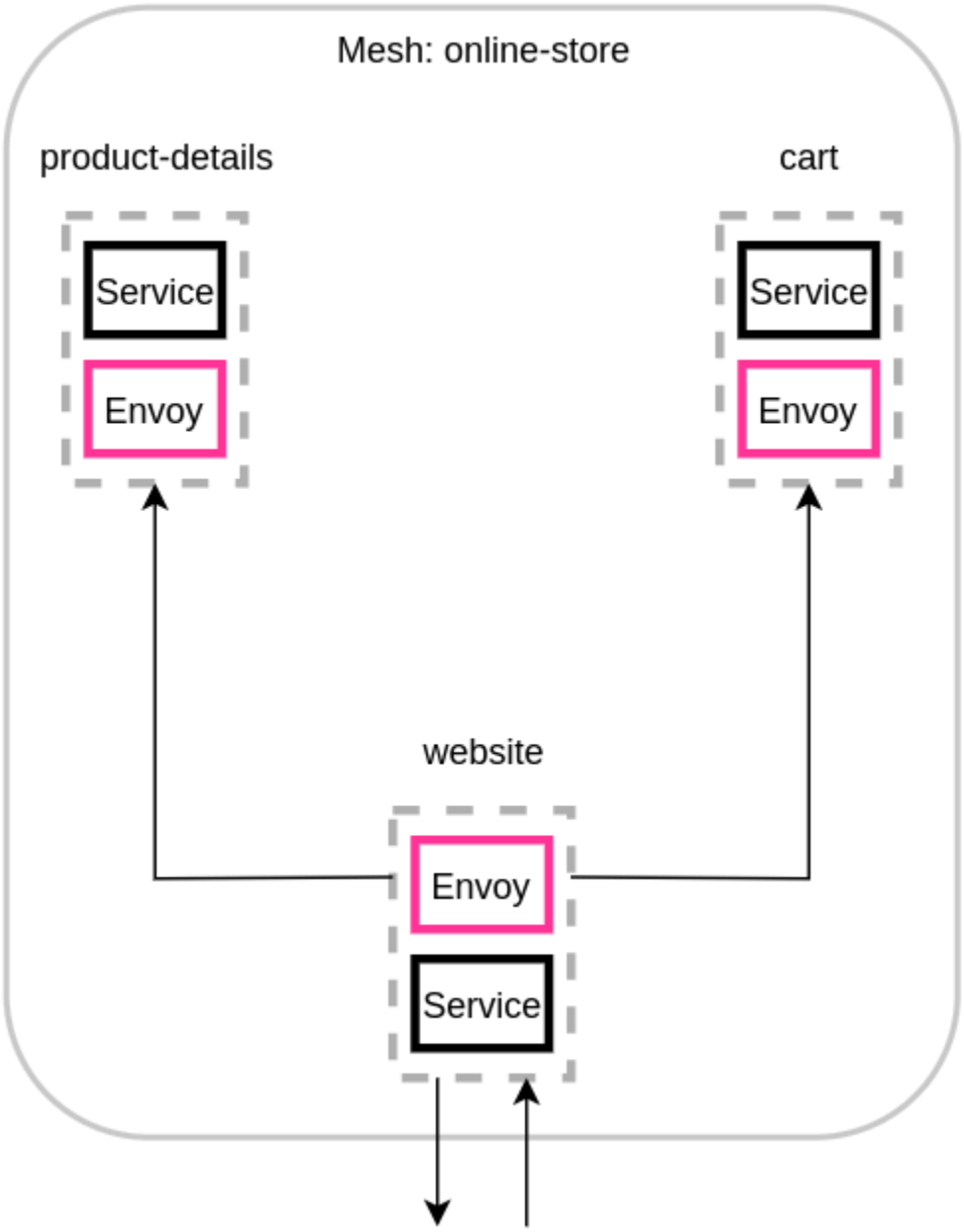
例如，如果您有名為 my-virtual-node的虛擬節點，在連接埠上接聽 http 流量，8080且該虛擬節點位於名為 的網格中my-mesh，App Mesh 會建立名為 的叢集cds_ingress_my-mesh_my-virtual-node_http_8080。此叢集可做為 my-virtual-node服務容器流量的目的地。

App Mesh 也可以建立下列類型的其他特殊叢集。這些其他叢集不一定對應至您在網格中明確定義的資源。

- 用於連接其他 AWS 服務的叢集。根據預設，此類型可讓您的網格到達大部分 AWS 的服務：cds_egress_*<mesh name>*_amazonaws。
- 用於為虛擬閘道執行路由的叢集。這通常可以安全忽略：。
  - 對於單一接聽程式： cds_ingress_*<mesh name>*_*<virtual gateway name>*_self_redirect_*<protocol>*_*<port>*
  - 對於多個接聽程式： cds_ingress_*<mesh name>*_*<virtual gateway name>*_self_redirect_*<ingress_listener_port>*_*<protocol>*_*<port>*
- 當您使用 Envoy 的 Secret Discovery Service 擷取秘密時，您可以定義端點的叢集，例如 TLS：static_cluster_sds_unix_socket。

應用程式指標範例

為了說明 Envoy 中可用的指標，下列範例應用程式有三個虛擬節點。可以忽略網格中的虛擬服務、虛擬路由器和路由，因為它們不會反映在 Envoy 的指標中。在此範例中，所有 服務都會接聽連接埠 8080 上的 http 流量。



我們建議您將環境變數新增至在網格中執行ENABLE_ENVOY_STATS_TAGS=1的 Envoy 代理容器。這會將下列指標維度新增至代理發出的所有指標：

- appmesh.mesh

- appmesh.virtual_node
- appmesh.virtual_gateway

這些標籤會設定為網格、虛擬節點或虛擬閘道的名稱，以允許使用網格中的資源名稱來篩選指標。

資源名稱

網站虛擬節點的代理具有下列資源：

- 輸入和輸出流量的兩個接聽程式：
  - lds_ingress_0.0.0.0_15000
  - lds_egress_0.0.0.0_15001
- 兩個輸出叢集，代表兩個虛擬節點後端：
  - cds_egress_online-store_product-details_http_8080
  - cds_egress_online-store_cart_http_8080
- 網站服務容器的傳入叢集：
  - cds_ingress_online-store_website_http_8080

接聽程式指標範例

- listener.0.0.0.0_15000.downstream_cx_active- 作用中輸入網路連線至 Envoy 的數目。
- listener.0.0.0.0_15001.downstream_cx_active—Envoy 的主動輸出網路連線數目。您的應用程式對外部服務的連線會包含在此計數中。
- listener.0.0.0.0_15000.downstream_cx_total- 輸入網路連線至 Envoy 的總數。
- listener.0.0.0.0_15001.downstream_cx_total- 輸出網路連線到 Envoy 的總數。

如需完整的接聽程式指標集，請參閱 Envoy 文件中的統計資料。

範例叢集指標

- cluster_manager.active_clusters- Envoy 已建立至少一個連線的叢集總數。
- cluster_manager.warming_clusters- Envoy 尚未連線的叢集總數。

下列叢集指標使用 的格式cluster.<cluster name>.<metric name>。這些指標名稱對應用程式範例而言是唯一的，並且由網站 Envoy 容器發出：

- cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_total- 網站與產品詳細資訊之間的連線總數。
- cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_connect_fail- 網站與產品詳細資訊之間的失敗連線總數。
- cluster.cds_egress_online-store_product-details_http_8080.health_check.failure- 網站與產品詳細資訊之間的失敗運作狀態檢查總數。
- cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_total- 在網站和產品詳細資訊之間提出的請求總數。
- cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_time- 網站與產品詳細資訊之間提出的請求所花費的時間。
- cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_2xx—網站從產品詳細資訊收到的 HTTP 2xx 回應數目。

如需完整的 HTTP 指標集，請參閱 Envoy 文件中的統計資料。

管理伺服器指標

Envoy 也會發出與 App Mesh 控制平面連線相關的指標，做為 Envoy 的管理伺服器。我們建議您監控其中一些指標，做為長時間從控制平面取消同步代理時通知您的方法。失去與控制平面的連線或更新失敗，會導致您的代理無法從 App Mesh 接收新組態，包括透過 App Mesh APIs進行的網格變更。

- control_plane.connected_state- 當代理連線到 App Mesh 時，此指標會設為 1，否則為 0。
- *.update_rejected—Envoy 拒絕的組態更新總數。這些通常是由於使用者設定錯誤所致。例如，如果您設定 App Mesh 從 Envoy 無法讀取的檔案讀取 TLS 憑證，則包含該憑證路徑的更新會遭到拒絕。
  - 對於已更新的接聽程式遭拒，統計資料將為 listener_manager.lds.update_rejected。
  - 對於已更新的叢集遭拒，統計資料將為 cluster_manager.cds.update_rejected。
- *.update_success—App Mesh 向代理成功進行組態更新的數量。這包括啟動新的 Envoy 容器時傳送的初始組態承載。
  - 對於接聽程式更新成功，統計資料將為 listener_manager.lds.update_success。
  - 對於叢集更新成功，統計資料將為 cluster_manager.cds.update_success。

如需管理伺服器指標集，請參閱 Envoy 文件中的管理伺服器。

匯出指標

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

Envoy 會針對自己的操作和傳入和傳出流量的各種維度發出許多統計資料。若要進一步了解 Envoy 統計資料，請參閱 Envoy 文件中的統計資料。這些指標可透過代理管理連接埠上的`/stats`端點取得，通常為 `9901`。

字`stat`首會有所不同，取決於您使用的是單一或多個接聽程式。以下是一些說明差異的範例。

> **Warning**
>
> 如果您將單一接聽程式更新為多個接聽程式功能，您可能會因為下表中說明的更新狀態字首而面臨重大變更。
>
> 我們建議您使用 Envoy 映像`1.22.2.1-prod`或更新版本。這可讓您在 Prometheus 端點中查看類似的指標名稱。

單一接聽程式 (SL)/具有「輸入」接聽程式字首的現有統計資料	多個接聽程式 (ML)/具有「輸入.<protocol>.<port>」接聽程式字首的新統計	

`http.*ingress*.rds.rds_ingress_http_5555.version_text`	`http.*ingress.http.5555*.rds.rds_ingress_http_5555.version_text`
	
	`http.*ingress.http.6666*.rds.rds_ingress_http_6666.version_text`	

單一接聽程式 (SL)/具有「輸入」接聽程式字首的現有統計資料	多個接聽程式 (ML)/具有「輸入.<protocol>.<port>」接聽程式字首的新統計	

`listener.0.0.0.0_15000.http.*ingress*.downstream_rq_2xx`	`listener.0.0.0.0_15000.http.*ingress.http.5555*.downstream_rq_2xx`
	
	`listener.0.0.0.0_15000.http.*ingress.http.6666*.downstream_rq_2xx`	

`http.*ingress*.downstream_cx_length_ms`	`http.*ingress.http.5555*.downstream_cx_length_ms`
	
	`http.*ingress.http.6666*.downstream_cx_length_ms`	

如需統計資料端點的詳細資訊，請參閱 Envoy 文件中的統計資料端點。如需管理介面的詳細資訊，請參閱 啟用 Envoy 代理管理介面。

搭配 Amazon EKS 的 App Mesh Prometheus

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

Prometheus 是一種開放原始碼監控和警示工具組。其功能之一是指定一個格式，以發出可供其他系統使用的指標。如需 Prometheus 的詳細資訊，請參閱 Prometheus 文件中的概觀。Envoy 可以透過傳入參數，透過其統計資料端點發出其指標`/stats?format=prometheus`。

對於使用 Envoy 映像建置 v1.22.2.1-prod 的客戶，有兩個額外的維度來表示輸入接聽程式特定統計資料：

- appmesh.listener_protocol
- appmesh.listener_port

以下是 Prometheus 現有統計資料與新統計資料之間的比較。

- 具有「輸入」接聽程式字首的現有統計資料

```
envoy_http_downstream_rq_xx{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_node="foodteller-
vn",envoy_response_code_class="2",envoy_http_conn_manager_prefix="ingress"} 931433
```

- 具有 "inress.<protocol>.<port>" + Appmesh Envoy Image v1.22.2.1-prod 或更新版本的新統計資料

```
envoy_http_downstream_rq_xx{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_node="foodteller-
vn",envoy_response_code_class="2",appmesh_listener_protocol="http",appmesh_listener_port="555
 20
```

- 具有 "inress.<protocol>.<port>" + 自訂 Envoy Imagebuild 的新統計資料

```
envoy_http_http_5555_downstream_rq_xx{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_node="foodteller-
vn",envoy_response_code_class="2",envoy_http_conn_manager_prefix="ingress"} 15983
```

對於多個接聽程式，cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>特殊叢集將是接聽程式特定的。

- 具有「輸入」接聽程式字首的現有統計資料

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_gateway="tellergateway-vg",Mesh="multiple-listeners-
mesh",VirtualGateway="tellergateway-vg",envoy_cluster_name="cds_ingress_multiple-
listeners-mesh_tellergateway-vg_self_redirect_http_15001"} 0
```

- 具有 "inress.<protocol>.<port>" 的新統計資料

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-
listeners-mesh",appmesh_virtual_gateway="tellergateway-
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_tellergateway-
vg_self_redirect_1111_http_15001"} 0
envoy_cluster_assignment_stale{appmesh_mesh="multiple-
listeners-mesh",appmesh_virtual_gateway="tellergateway-
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_tellergateway-
vg_self_redirect_2222_http_15001"} 0
```

安裝 Prometheus

1. 將 EKS 儲存庫新增至 Helm：

```
helm repo add eks https://aws.github.io/eks-charts
```

2. 安裝 App Mesh Prometheus

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system
```

Prometheus 範例

以下是`PersistentVolumeClaim`為 Prometheus 持久性儲存體建立 的範例。

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system \
--set retention=12h \
--set persistentVolumeClaim.claimName=prometheus
```

使用 Prometheus 的逐步解說

- 具有 EKS 的應用程式網格 - 可觀測性：Prometheus

進一步了解 Prometheus 和 Prometheus 搭配 Amazon EKS

- Prometheus 文件
- EKS - 使用 Prometheus 控制平面指標

適用於 App Mesh 的 CloudWatch

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

從 Amazon EKS 將 Envoy 統計資料傳送至 CloudWatch

您可以將 CloudWatch Agent 安裝到您的叢集，並設定它以從您的代理收集一部分指標。如果您還沒有 Amazon EKS 叢集，則可以使用逐步解說：在 GitHub 上使用 Amazon EKS 的應用程式網格中的步驟來建立叢集。 GitHub 您可以依照相同的演練，將範例應用程式安裝到叢集。

若要為您的叢集設定適當的 IAM 許可並安裝代理程式，請遵循安裝 CloudWatch Agent with Prometheus Metrics Collection 中的步驟。預設安裝包含 Prometheus 抓取組態，可提取 Envoy 統計資料的有用子集。如需詳細資訊，請參閱 App Mesh 的 Prometheus 指標。

若要建立 App Mesh 自訂 CloudWatch 儀表板，其設定為顯示代理程式收集的指標，請遵循檢視您的 Prometheus 指標教學課程中的步驟。當流量進入 App Mesh 應用程式時，您的圖形會開始填入對應的指標。

篩選 CloudWatch 的指標

App Mesh 指標延伸提供有用的指標子集，可讓您深入了解您在網格中定義之資源的行為。由於 CloudWatch 代理程式支援抓取 Prometheus 指標，因此您可以提供抓取組態，以選取要從 Envoy 提取並傳送至 CloudWatch 的指標。

您可以在我們的指標延伸演練中找到使用 Prometheus 抓取指標的範例。

CloudWatch 範例

您可以在我們的範例AWS 儲存庫中找到 CloudWatch 的範例組態。

使用 CloudWatch 的逐步解說

- 在我們的 App Mesh 研討會中新增監控和記錄功能。
- 具有 EKS 的應用程式網格 - 可觀測性：CloudWatch
- 在 ECS 上使用 App Mesh 的指標延伸

App Mesh 的指標延伸

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

Envoy 會產生數百個指標，細分為幾個不同的維度。這些指標與 App Mesh 的關係並不直接。在虛擬服務的情況下，沒有機制可以確定哪個虛擬服務正在與指定的虛擬節點或虛擬閘道通訊。

App Mesh 指標延伸功能可增強在網格中執行的 Envoy 代理。此增強功能可讓代理發出其他指標，這些指標會知道您定義的資源。這個小部分的其他指標將協助您更深入地了解您在 App Mesh 中定義的這些資源的行為。

若要啟用 App Mesh 指標延伸，請將環境變數設定為 `APPMESH_METRIC_EXTENSION_VERSION` 1。

```
APPMESH_METRIC_EXTENSION_VERSION=1
```

如需 Envoy 組態變數的詳細資訊，請參閱Envoy 組態變數。

與傳入流量相關的指標

- **`ActiveConnectionCount`**
  - `envoy.appmesh.ActiveConnectionCount` — 作用中 TCP 連線的數量。
  - 維度 — Mesh、VirtualNode、VirtualGateway
- **`NewConnectionCount`**
  - `envoy.appmesh.NewConnectionCount` — TCP 連線總數。
  - 維度 — Mesh、VirtualNode、VirtualGateway
- **`ProcessedBytes`**

  - envoy.appmesh.ProcessedBytes — 從下游用戶端傳送和接收的 TCP 位元組總數。
  - 維度 — Mesh、VirtualNode、VirtualGateway
- **RequestCount**
  - envoy.appmesh.RequestCount — 已處理的 HTTP 請求數目。
  - 維度 — Mesh、VirtualNode、VirtualGateway
- **GrpcRequestCount**
  - envoy.appmesh.GrpcRequestCount — 已處理的 gPRC 請求數目。
  - 維度 — Mesh、VirtualNode、VirtualGateway

與傳出流量相關的指標

根據輸出指標來自虛擬節點或虛擬閘道，您會看到不同的維度。

- **TargetProcessedBytes**
  - envoy.appmesh.TargetProcessedBytes — 傳送至和接收自 Envoy 上游目標的 TCP 位元組總數。
  - 維度：
    - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
    - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode
- **HTTPCode_Target_2XX_Count**
  - envoy.appmesh.HTTPCode_Target_2XX_Count — 對 Envoy 上游目標發出導致 2xx HTTP 回應的 HTTP 請求數量。
  - 維度：
    - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
    - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode
- **HTTPCode_Target_3XX_Count**
  - envoy.appmesh.HTTPCode_Target_3XX_Count — 對 Envoy 上游目標發出導致 3xx HTTP 回應的 HTTP 請求數量。
  - 維度：
    - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
    - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode
- **HTTPCode_Target_4XX_Count**

- `envoy.appmesh.HTTPCode_Target_4XX_Count` — 對 Envoy 上游目標發出導致 4xx HTTP 回應的 HTTP 請求數量。
- 維度：
  - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
  - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode
- **`HTTPCode_Target_5XX_Count`**
  - `envoy.appmesh.HTTPCode_Target_5XX_Count` — 對 Envoy 上游目標發出導致 5xx HTTP 回應的 HTTP 請求數量。
  - 維度：
    - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
    - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode
- **`RequestCountPerTarget`**
  - `envoy.appmesh.RequestCountPerTarget` — 傳送至 Envoy 上游目標的請求數量。
  - 維度：
    - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
    - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode
- **`TargetResponseTime`**
  - `envoy.appmesh.TargetResponseTime` — 從向 Envoy 上游的目標發出請求到收到完整回應所經過的時間。
  - 維度：
    - 虛擬節點維度 — Mesh、VirtualNode、TargetVirtualService、TargetVirtualNode
    - 虛擬閘道維度 — Mesh、VirtualGateway、TargetVirtualService、TargetVirtualNode

App Mesh 的 Datadog

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

Datadog 是一種監控和安全應用程式，用於雲端應用程式的端對端監控、指標和記錄。Datadog 可讓您的基礎設施、應用程式和第三方應用程式完全可見。

安裝 Datadog

- EKS - 若要使用 EKS 設定 Datadog，請遵循 Datadog 文件中的下列步驟。
- ECS EC2 - 若要使用 ECS EC2 設定 Datadog，請遵循 Datadog 文件中的下列步驟。

若要進一步了解 Datadog

- Datadog 文件

追蹤

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

> **Important**
>
> 若要完全實作追蹤，您需要更新您的應用程式。
> 若要查看所選服務中的所有可用資料，您必須使用適用的程式庫來檢測應用程式。

使用 AWS X-Ray 監控應用程式網格

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS X-Ray 是一種服務，提供工具，可讓您檢視、篩選和深入了解從應用程式提供的請求所收集的資料。這些洞見可協助您識別問題和機會，以最佳化您的應用程式。您可以查看請求和回應的詳細資訊，以及應用程式對其他服務 AWS 進行的下游呼叫。

X-Ray 與 App Mesh 整合，以管理您的 Envoy 微服務。來自 Envoy 的追蹤資料會傳送到您容器中執行的 X-Ray 協助程式。

使用您語言專用的 SDK 指南，在您的應用程式程式碼中實作 X-Ray。

透過 App Mesh 啟用 X-Ray 追蹤

- 視服務類型而定：
  - ECS - 在 Envoy 代理容器定義中，將`ENABLE_ENVOY_XRAY_TRACING`環境變數設定為 `1`，並將`XRAY_DAEMON_PORT`環境變數設定為 `2000`。
  - EKS - 在應用程式網格控制器組態中，包含 `--set tracing.enabled=true`和 `--set tracing.provider=x-ray`。
- 在您的 X-Ray 容器中，公開連接埠`2000`並以使用者 身分執行`1337`。

X-Ray 範例

Amazon ECS 的 Envoy 容器定義

```
      {
        "name": "envoy",
        "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
        "essential": true,
        "environment": [
          {
            "name": "APPMESH_VIRTUAL_NODE_NAME",
            "value": "mesh/myMesh/virtualNode/myNode"
          },
          {
            "name": "ENABLE_ENVOY_XRAY_TRACING",
            "value": "1"
           }
        ],
        "healthCheck": {
          "command": [
```

```
            "CMD-SHELL",
            "curl -s http://localhost:9901/server_info | cut -d' ' -f3 | grep -q live"
            ],
          "startPeriod": 10,
          "interval": 5,
          "timeout": 2,
          "retries": 3
      }
```

更新 Amazon EKS 的 App Mesh 控制器

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
--namespace appmesh-system \
--set region=${AWS_REGION} \
--set serviceAccount.create=false \
--set serviceAccount.name=appmesh-controller \
--set tracing.enabled=true \
--set tracing.provider=x-ray
```

使用 X-Ray 的逐步解說

- 使用 AWS X-Ray 監控
- 搭配 Amazon EKS 的應用程式網格 - 可觀測性：X-Ray
- 在 AWS App Mesh Workshop 中使用 X-Ray 進行分散式追蹤

進一步了解 AWS X-Ray

- AWS X-Ray 文件

使用 App Mesh 對 AWS X-Ray 進行故障診斷

- 無法查看我應用程式的 AWS X-Ray 追蹤。

搭配 Amazon EKS 的 App Mesh 的 Jaeger

Jaeger 是開放原始碼的端對端分散式追蹤系統。它可用於描述網路和 的監控。Jaeger 也可以協助您疑難排解複雜的雲端原生應用程式。

若要在應用程式程式碼中實作 Jaeger，您可以在 Jaeger 文件追蹤程式庫中找到您語言特定的指南。

使用 Helm 安裝 Jaeger

1. 將 EKS 儲存庫新增至 Helm：

```
helm repo add eks https://aws.github.io/eks-charts
```

2. 安裝 App Mesh Jaeger

```
helm upgrade -i appmesh-jaeger eks/appmesh-jaeger \
--namespace appmesh-system
```

Jaeger 範例

以下是`PersistentVolumeClaim`為 Jaeger 持久性儲存建立 的範例。

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
--namespace appmesh-system \
--set tracing.enabled=true \
--set tracing.provider=jaeger \
--set tracing.address=appmesh-jaeger.appmesh-system \
--set tracing.port=9411
```

使用 Jaeger 的逐步解說

- 具有 EKS 的應用程式網格 - 可觀測性：Jaeger

若要進一步了解 Jaeger

- Jaeger 文件

用於追蹤的資料狗

Datadog 可用於追蹤和指標。如需詳細資訊和安裝說明，請參閱 Datadog 文件中的應用程式語言專屬指南。

App Mesh 工具

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

App Mesh 可讓客戶使用下列工具間接與其 APIs互動：

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- Kubernetes 的應用程式網格控制器
- Terraform

應用程式網格和 AWS CloudFormation

AWS CloudFormation 是一項服務，可讓您建立範本，其中包含應用程式所需的所有資源，然後 AWS CloudFormation 會為您設定和佈建資源。它也會設定所有相依性，因此您可以更專注於您的應用程式，並減少管理資源的作業。

如需 AWS CloudFormation 搭配 App Mesh 使用 的詳細資訊和範例，請參閱 AWS CloudFormation 文件。

應用程式網格和 AWS CDK

AWS CDK 是使用程式碼來定義雲端基礎設施並使用 AWS CloudFormation 來佈建雲端基礎設施的開發架構。 AWS CDK 支援多種程式設計語言，包括 TypeScript、JavaScript、Python、Java 和 C#/。淨額。

如需 AWS CDK 搭配 App Mesh 使用 的詳細資訊，請參閱 AWS CDK 文件。

適用於 Kubernetes 的 App Mesh 控制器

Kubernetes 的 App Mesh 控制器可協助您管理 Kubernetes 叢集的 App Mesh 資源，並將附屬項目注入 Pod。此控制器專門與 Amazon EKS 搭配使用，可讓您以 Kubernetes 原生的方式管理資源。

如需 App Mesh 控制器的詳細資訊，請參閱 App Mesh 控制器文件。

App Mesh 和 Terraform

Terraform 是一種開放原始碼基礎設施，做為程式碼軟體工具。Terraform 可以使用更精簡的 CLI 管理雲端服務，並使用明確組態檔案與 APIs 互動。

若要進一步了解如何搭配 Terraform 使用 App Mesh，請參閱 Terraform 文件。

使用共用網格

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

您可以使用 AWS Resource Access Manager 服務跨 AWS 帳戶共用您的 App Mesh 網格。共用網格可讓不同 AWS 帳戶建立的資源在同一個網格中彼此通訊。

AWS 帳戶可以是網格資源擁有者、網格取用者或兩者。消費者可以在與其帳戶共用的網格中建立資源。擁有者可以在帳戶擁有的任何網格中建立資源。網格擁有者可以與下列類型的網格取用者共用網格。

- 中組織內部或外部的特定 AWS 帳戶 AWS Organizations
- 中的組織單位 AWS Organizations
- 其整個組織位於 AWS Organizations

如需共用網格的end-to-end逐步解說，請參閱 GitHub 上的跨帳戶網格逐步解說。

授予許可以共用網格

在帳戶之間共用網格時，IAM 主體需要許可才能共用網格，以及網格本身所需的資源層級許可。

授予共用網格的許可

IAM 主體共享網格需要一組最低許可。我們建議您使用 `AWSAppMeshFullAccess`和 `AWSResourceAccessManagerFullAccess`受管 IAM 政策，以確保您的 IAM 主體擁有共用和使用共用網格所需的許可。

如果您使用自訂 IAM 政策，則需要 `appmesh:PutMeshPolicy`、 `appmesh:GetMeshPolicy`和 `appmesh:DeleteMeshPolicy`動作。這些是僅限許可的 IAM 動作。如果 IAM 主體未授予這些許可，則嘗試使用 AWS RAM 服務共用網格時將發生錯誤。

如需 AWS Resource Access Manager 服務使用 IAM 方式的詳細資訊，請參閱AWS Resource Access Manager 《 使用者指南》中的AWS RAM 如何使用 IAM。

授予網格的許可

共用網格具有下列許可。

- 消費者可以列出和描述與帳戶共用的網格中的所有資源。
- 擁有者可以列出和描述帳戶擁有的任何網格中的所有資源。
- 擁有者和消費者可以修改帳戶建立的網格中的資源，但無法修改其他帳戶建立的資源。
- 取用者可以刪除帳戶建立的網格中的任何資源。
- 擁有者可以刪除網格中任何帳戶建立的任何資源。
- 擁有者的資源只能參考相同帳戶中的其他資源。例如，虛擬節點只能參考 AWS Cloud Map 或與 AWS Certificate Manager 虛擬節點擁有者位於相同帳戶的憑證。
- 擁有者和消費者可以將 Envoy 代理連接到 App Mesh，做為帳戶擁有的虛擬節點。
- 擁有者可以建立虛擬閘道和虛擬閘道路由。
- 擁有者和消費者可以列出標籤，並可以在帳戶建立的網格中標記/取消標記資源。它們無法列出非由帳戶建立之網格中的標籤和標籤/取消標籤資源。

共用網格使用以政策為基礎的授權。網格與固定的一組許可共用。選取要新增到資源政策的這些許可，也可以根據 IAM 使用者/角色選取選用的 IAM 政策。這些政策中允許的許可交集，減去任何拒絕的明確許可，會決定主體對網格的存取。

共用網格時， AWS Resource Access Manager 服務會建立名為 的受管政策，`AWSRAMDefaultPermissionAppMesh`並將其與提供下列許可的 App Mesh 建立關聯。

- `appmesh:CreateVirtualNode`
- `appmesh:CreateVirtualRouter`
- `appmesh:CreateRoute`
- `appmesh:CreateVirtualService`
- `appmesh:UpdateVirtualNode`
- `appmesh:UpdateVirtualRouter`
- `appmesh:UpdateRoute`
- `appmesh:UpdateVirtualService`
- `appmesh:ListVirtualNodes`
- `appmesh:ListVirtualRouters`

- appmesh:ListRoutes
- appmesh:ListVirtualServices
- appmesh:DescribeMesh
- appmesh:DescribeVirtualNode
- appmesh:DescribeVirtualRouter
- appmesh:DescribeRoute
- appmesh:DescribeVirtualService
- appmesh:DeleteVirtualNode
- appmesh:DeleteVirtualRouter
- appmesh:DeleteRoute
- appmesh:DeleteVirtualService
- appmesh:TagResource
- appmesh:UntagResource

共用網格的先決條件

若要共用網格，您必須符合下列先決條件。

- 您必須擁有 AWS 帳戶中的網格。您無法共用已與您共用的網格。
- 若要與組織或 中的組織單位共用網格 AWS Organizations，您必須啟用與 共用 AWS Organizations。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的透過 AWS Organizations啟用共用。
- 您的服務必須部署在 Amazon VPC 中，該 VPC 具有跨帳戶共用連線，其中包含您要彼此通訊的網格資源。共用網路連線的其中一種方法是將要在網格中使用的所有服務部署到共用子網路。如需詳細資訊和限制，請參閱共用子網路。
- 服務必須透過 DNS 或 進行探索 AWS Cloud Map。如需服務探索的詳細資訊，請參閱虛擬節點。

相關服務

網格共享與 AWS Resource Access Manager (AWS RAM) 整合。 AWS RAM 是一項服務，可讓您與任何 AWS 帳戶或透過 共享 AWS 資源 AWS Organizations。您可以使用 AWS RAM建立資源共享，以共享您擁有的資源。資源共享指定要共用的資源，以及共用它們的消費者。消費者可以是個別 AWS 帳戶、組織單位或整個組織 AWS Organizations。

如需詳細資訊 AWS RAM，請參閱 AWS RAM 使用者指南。

共用網格

共用網格可讓不同帳戶建立的網格資源在同一個網格中彼此通訊。您只能共用您擁有的網格。若要共用網格，您必須將其新增至資源共用。資源共用是一種 AWS RAM 資源，可讓您跨 AWS 帳戶共用資源。資源共用會指定要共用的資源，以及與其共用的消費者。當您使用 Amazon Linux 主控台共用網格時，您可以將其新增至現有的資源共用。若要將網格新增至新的資源共享，請使用 AWS RAM 主控台建立資源共享。

如果您是 中組織的一部分， AWS Organizations 且已啟用組織內的共用，則組織中的消費者可以自動獲得共用網格的存取權。否則，消費者會收到加入資源共享的邀請，並在接受邀請後授予共用網格的存取權。

您可以使用 AWS RAM 主控台或 共享您擁有的網格 AWS CLI。

使用 AWS RAM 主控台共用您擁有的網格

如需說明，請參閱AWS RAM 《 使用者指南》中的建立資源共用。選取資源類型時，請選取網格，然後選取您要共用的網格。如果未列出網格，請先建立網格。如需詳細資訊，請參閱建立服務網格。

使用 共享您擁有的網格 AWS CLI

使用 create-resource-share 命令。針對 `--resource-arns`選項，指定您要共用之網格的 ARN。

取消共用共用網格

當您取消共用網格時，App Mesh 會停用網格的前消費者對網格的進一步存取。不過，App Mesh 不會刪除消費者建立的資源。取消共用網格之後，只有網格擁有者可以存取和刪除資源。App Mesh 可防止在網格中擁有資源的帳戶在未共用網格後接收組態資訊。App Mesh 也可防止具有網格中資源的任何其他帳戶從未共用的網格接收組態資訊。只有網格的擁有者可以取消共用。

若要取消共用您擁有的共用網格，您必須將其從資源共用中移除。您可以使用 AWS RAM 主控台或 來執行此操作 AWS CLI。

使用 AWS RAM 主控台取消共用您擁有的共用網格

如需說明，請參閱AWS RAM 《 使用者指南》中的更新資源共用。

使用 取消共用您擁有的共用網格 AWS CLI

使用 disassociate-resource-share 命令。

識別共用網格

擁有者和消費者可以使用 Amazon Linux 主控台和 識別共用網格和網格資源 AWS CLI

使用 Amazon Linux 主控台識別共用網格

1. 在 https：//https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 從左側導覽中，選取網格。每個網格的網格擁有者帳戶 ID 會列在網格擁有者欄中。
3. 從左側導覽中，選取虛擬服務、虛擬路由器或虛擬節點。您可以看到網格擁有者的帳戶 ID，以及每個資源的資源擁有者。

使用 識別共用網格 AWS CLI

使用 `aws appmesh list resource`命令，例如 `aws appmesh list-meshes`。命令會傳回您擁有的網格，以及與您共用的網格。`meshOwner` 屬性會顯示 AWS 的帳戶 ID，`meshOwner`而 `resourceOwner` 屬性會顯示資源擁有者 AWS 的帳戶 ID。針對任何網格資源執行的任何命令都會傳回這些屬性。

您連接到共用網格的使用者定義標籤僅適用於您的 AWS 帳戶。它們不適用於與網格共用的其他帳戶。另一個帳戶中網格的 `aws appmesh list-tags-for-resource` 命令被拒絕存取。

計費和計量

共用網格不收取任何費用。

執行個體配額

網格的所有配額也適用於共用網格，無論網格中誰建立了資源。只有網格擁有者可以請求增加配額。如需詳細資訊，請參閱App Mesh 服務配額。 AWS Resource Access Manager 服務也有配額。如需詳細資訊，請參閱 Service Quotas。

AWS 與 App Mesh 整合的 服務

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

App Mesh 與其他 AWS 服務搭配使用，為您的業務挑戰提供額外的解決方案。本主題識別使用 App Mesh 來新增功能的 服務，或 App Mesh 用來執行任務的服務。

目錄

使用 建立 App Mesh 資源 AWS CloudFormation

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

App Mesh 已與 整合 AWS CloudFormation，這項服務可協助您建立和設定 AWS 資源的模型，讓您可減少建立和管理資源和基礎設施的時間。您可以建立範本來描述您想要的所有 AWS 資源，例如 App Mesh 網格，並 AWS CloudFormation 負責為您佈建和設定這些資源。

使用 時 AWS CloudFormation，您可以重複使用範本，以一致且重複地設定 App Mesh 資源。只需描述您的資源一次，然後在多個 AWS 帳戶和區域中逐一佈建相同的資源。

應用程式網格和 AWS CloudFormation 範本

若要佈建和設定 App Mesh 和相關服務的資源，您必須了解 AWS CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您想要在 AWS CloudFormation 堆疊中佈建的資

源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation 設計工具來協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的什麼是 AWS CloudFormation 設計工具？。

App Mesh 支援在其中建立網格、路由、虛擬節點、虛擬路由器和虛擬服務 AWS CloudFormation。如需詳細資訊，包括您 App Mesh 資源的 JSON 和 YAML 範本範例，請參閱AWS CloudFormation 《 使用者指南》中的 App Mesh 資源類型參考。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- AWS CloudFormation 命令列界面使用者指南

AWS Outposts 上的應用程式網格

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS Outposts 可在內部部署設施中啟用原生 AWS 服務、基礎設施和操作模型。在 AWS Outposts 環境中，您可以使用與 AWS 雲端中使用的相同 AWS APIs、工具和基礎設施。App Mesh on AWS Outposts 非常適合需要在接近現場部署資料和應用程式的情況下執行的低延遲工作負載。如需 AWS Outposts 的詳細資訊，請參閱 AWS Outposts 使用者指南。

先決條件

以下是在 AWS Outpost 上使用 App Mesh 的先決條件：

- 內部部署資料中心必須已安裝和設定 Outpost。
- Outpost 與其 AWS 區域之間必須有可靠的網路連線。
- Outpost AWS 的區域必須支援 AWS App Mesh。如需支援區域的清單，請參閱 中的AWS App Mesh 端點和配額AWS 一般參考。

限制

以下是在 AWS Outposts 上使用 App Mesh 的限制：

- AWS Identity and Access Management、Application Load Balancer、Network Load Balancer、Classic Load Balancer 和 Amazon Route 53 會在 AWS 區域中執行，而不是在 Outposts 上執行。這將提高這些服務和容器之間的延遲。

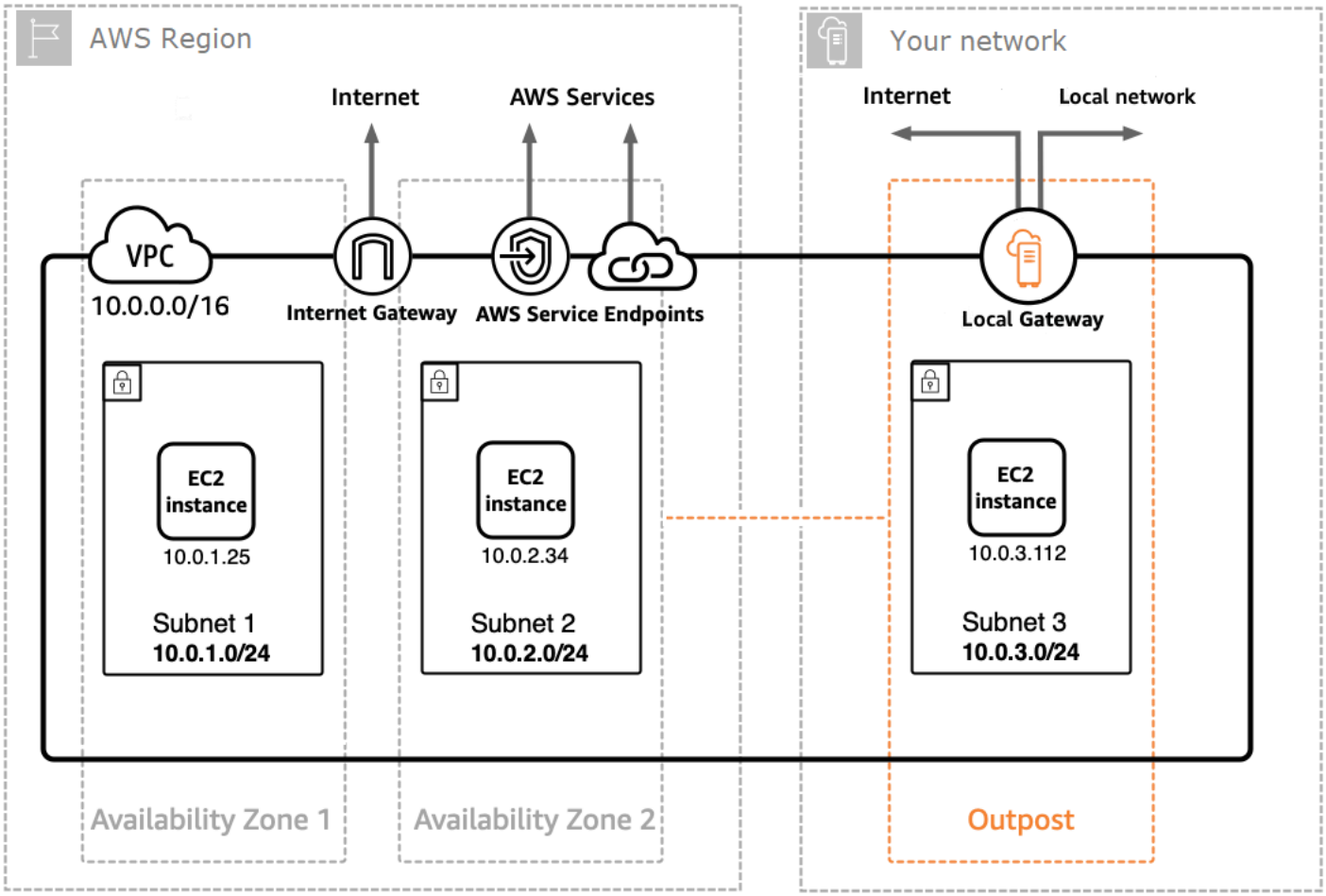
網路連線能力考量

以下是 Amazon EKS AWS Outposts 的網路連線考量：

- 如果您的 Outpost 與其 AWS 區域之間的網路連線中斷，App Mesh Envoy 代理將繼續執行。不過，在恢復連線之前，您將無法修改服務網格。
- 我們建議您在 Outpost 及其 AWS 區域之間提供可靠、高可用性和低延遲的連線。

在 Outpost 上建立 App Mesh Envoy 代理

Outpost 是 AWS 區域的延伸，您可以在 帳戶中擴展 Amazon VPC，以跨越多個可用區域和任何相關聯的 Outpost 位置。當您設定 Outpost 時，您會將子網路與之相關聯，使您的區域性 VPC 環境延伸到內部部署設施。Outpost 上的執行個體會顯示為區域 VPC 的一部分，類似於具有相關子網路的可用區域。



若要在 Outpost 上建立 App Mesh Envoy 代理，請將 App Mesh Envoy 容器映像新增至在 Outpost 上執行的 Amazon ECS 任務或 Amazon EKS Pod。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 Amazon Elastic Container Service on AWS Outposts 和》Amazon EKS 使用者指南》中的 Amazon Elastic Kubernetes Service on AWS Outposts。

App Mesh 最佳實務

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

為了實現在計劃部署期間和部分主機意外遺失期間，零失敗請求的目標，本主題的最佳實務會實作下列策略：

- 使用安全的預設重試策略，從應用程式的角度提高請求成功的可能性。如需詳細資訊，請參閱使用重試檢測所有路由。
- 透過將重試請求傳送至實際目的地的可能性最大化，提高重試請求成功的可能性。如需詳細資訊，請參閱調整部署速度、在向內擴展之前向外擴展及實作容器運作狀態檢查。

若要大幅減少或消除故障，建議您在下列所有實務中實作建議。

使用重試檢測所有路由

將所有虛擬服務設定為使用虛擬路由器，並為所有路由設定預設重試政策。這將透過重新選取主機並傳送新請求來緩解失敗的請求。對於重試政策，我們建議 的值至少為 2`maxRetries`，並為每個支援重試事件類型的路由類型中的每種重試事件類型指定下列選項：

- TCP – `connection-error`
- HTTP 和 HTTP/2 – `stream-error`和 `gateway-error`
- gRPC – `cancelled`和 `unavailable`

其他重試事件需要逐case-by-case考慮，因為它們可能不安全，例如請求不等冪。您將需要考慮和測試 `maxRetries`和 的值`perRetryTimeout`，以適當權衡請求的最大延遲 (`maxRetries` * `perRetryTimeout`) 與增加更多重試的成功率。此外，當 Envoy 嘗試連線到不再存在的端點時，您應該預期該請求會使用完整的 `perRetryTimeout`。若要設定重試政策，請參閱 建立路由，然後選取您要路由的通訊協定。

> **Note**
>
> 如果您在 2020 年 7 月 29 日或之後實作路由，但未指定重試政策，則 App Mesh 可能已針對您在 2020 年 7 月 29 日或之後建立的每個路由，自動建立與先前政策類似的預設重試政策。如需詳細資訊，請參閱預設路由重試政策。

調整部署速度

使用滾動部署時，請降低整體部署速度。根據預設，Amazon ECS 會設定至少 100% 運作狀態良好的任務和 200% 總任務的部署策略。部署時，這會導致兩個高度偏離點：

- 在準備完成請求之前，Envoys 可能會看到新任務的 100% 機群大小（請參閱 實作容器運作狀態檢查 了解緩解措施）。
- 在任務終止時，Envoys 可能會看到舊任務的 100% 機群大小。

設定這些部署限制時，容器協調程式可能會進入一個狀態，同時隱藏所有舊目的地並顯示所有新目的地。由於您的 Envoy 資料平面最終一致，這可能會導致資料平面中可見的一組目的地與協調者的觀點不同。若要緩解這種情況，我們建議至少維持 100% 正常運作的任務，但將任務總數降低到 125%。這將減少分歧並改善重試的可靠性。針對不同的容器執行時間，我們建議執行下列設定：

Amazon ECS

如果您的服務需要計數為 2 或 3，請將 `maximumPercent`設為 150%。否則，請將 `maximumPercent`設為 125%。

Kubernetes

將部署的 設定為 `maxUnavailable` 0% `update strategy`和 `maxSurge` 25%。如需部署的詳細資訊，請參閱 Kubernetes 部署文件。

在向內擴展之前向外擴展

向外擴展和向內擴展都可能導致重試中失敗請求的一些可能性。雖然有一些任務建議可以減少橫向擴展，但向內擴展的唯一建議是將任務中的橫向擴展百分比降至最低。我們建議您在擴展舊任務或部署之前，使用擴展新 Amazon ECS 任務或 Kubernetes 部署的部署策略。此擴展策略可讓您在任務或部署中擴展的百分比保持較低，同時維持相同的速度。此實務同時適用於 Amazon ECS 任務和 Kubernetes 部署。

實作容器運作狀態檢查

在擴展案例中，Amazon ECS 任務中的容器可能會變得不按順序出現，並且最初可能沒有回應。對於不同的容器執行期，我們建議下列建議：

Amazon ECS

為了緩解這種情況，我們建議使用容器運作狀態檢查和容器相依性排序，以確保 Envoy 在需要開始傳出網路連線的任何容器之前，執行正常。若要在任務定義中正確設定應用程式容器和 Envoy 容器，請參閱容器相依性。

Kubernetes

無，因為在 Kubernetes 的 App Mesh 控制器中註冊和取消註冊 AWS Cloud Map 執行個體時，不會考慮 Kubernetes 存活和整備探查。 https://github.com/aws/aws-app-mesh-controller-for-k8s如需詳細資訊，請參閱 GitHub 問題 #132。

最佳化 DNS 解析度

如果您使用 DNS 進行服務探索，則必須選取適當的 IP 通訊協定，以在設定網格時最佳化 DNS 解析度。App Mesh 同時支援 IPv4和 IPv6，您選擇的 可能會影響服務的效能和相容性。如果您的基礎設施不支援 IPv6，我們建議您指定符合基礎設施的 IP 設定，而不是依賴預設IPv6_PREFERRED行為。預設IPv6_PREFERRED行為可能會降低服務效能。

- IPv6_PREFERRED – 這是預設設定。Envoy 會先執行 IPv6 地址的 DNS 查詢，IPv4如果找不到IPv6地址，則會回到 。如果您的基礎設施主要支援 IPv6，但需要IPv4相容性，這會很有幫助。
- IPv4_PREFERRED – 如果沒有可用的IPv4地址IPv6，Envoy 會先查詢IPv4地址，然後回到 。如果您的基礎設施主要支援 IPv4，但有一些IPv6相容性，請使用此設定。
- IPv6_ONLY – 如果您的服務僅支援IPv6流量，請選擇此選項。Envoy 只會對IPv6地址執行 DNS 查詢，確保所有流量都透過 路由IPv6。
- IPv4_ONLY – 如果您的服務僅支援IPv4流量，請選擇此設定。Envoy 只會對IPv4地址執行 DNS 查詢，確保所有流量都透過 路由IPv4。

您可以在網格層級和虛擬節點層級設定 IP 版本偏好設定，虛擬節點設定則會覆寫網格層級的 IP 版本偏好設定。

如需詳細資訊，請參閱服務網格和虛擬節點。

中的安全性 AWS App Mesh

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請造訪此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

的雲端安全 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全是 AWS 與您之間的共同責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 Cloud AWS 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要進一步瞭解適用於 AWS App Mesh的合規計劃，請參閱 合規計劃範圍內的AWS 服務。App Mesh 負責安全地將組態交付至本機代理，包括 TLS 憑證私有金鑰等秘密。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須負責其他因素，包括：
  - 資料的機密性、您公司的要求，以及適用的法律和法規。
  - App Mesh 資料平面的安全組態，包括允許流量在 VPC 內的服務之間傳遞的安全群組組態。
  - 與 App Mesh 相關聯的運算資源組態。
  - 與您的運算資源相關聯的 IAM 政策，以及它們可以從 App Mesh 控制平面擷取的組態。

本文件可協助您了解如何在使用 App Mesh 時套用共同責任模型。下列主題說明如何設定 App Mesh 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 App Mesh 資源。

App Mesh 安全性原則

客戶應該能夠視需要調整安全性。平台不應阻止它們更安全。根據預設，平台功能是安全的。

主題

- Transport Layer Security (TLS)
- 交互 TLS 驗證

- AWS App Mesh 如何使用 IAM
- 使用 記錄 AWS App Mesh API 呼叫 AWS CloudTrail
- 中的資料保護 AWS App Mesh
- 的合規驗證 AWS App Mesh
- 中的基礎設施安全性 AWS App Mesh
- 中的彈性 AWS App Mesh
- 中的組態和漏洞分析 AWS App Mesh

Transport Layer Security (TLS)

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

在 App Mesh 中，Transport Layer Security (TLS) 會加密部署在運算資源上的 Envoy 代理程式之間的通訊，這些資源由網格端點表示，例如 虛擬節點和 虛擬閘道。代理會交涉和終止 TLS。使用應用程式部署代理時，您的應用程式程式碼不負責交涉 TLS 工作階段。代理會代表您的應用程式交涉 TLS。

App Mesh 可讓您以下列方式提供 TLS 憑證給代理：

- 來自 AWS Certificate Manager (ACM) 的私有憑證，由 () AWS Private Certificate Authority 發行 AWS Private CA。
- 儲存在虛擬節點本機檔案系統中的憑證，由您自己的憑證授權機構 (CA) 發行
- Secrets Discovery Service (SDS) 端點透過本機 Unix Domain Socket 提供的憑證。

Envoy Proxy 授權 必須針對網格端點所代表的已部署 Envoy 代理啟用。我們建議您在啟用代理授權時，限制只能存取您要啟用加密的網格端點。

憑證需求

憑證上的其中一個主體別名 (SANs) 必須符合特定條件，取決於網格端點所代表的實際服務如何被探索。

- DNS – 其中一個憑證 SANs 必須符合 DNS 服務探索設定中提供的值。對於具有服務探索名稱 的應用程式*mesh-endpoint.apps.local*，您可以建立與該名稱相符的憑證，或使用萬用字元 的憑證*.*apps.local*。
- AWS Cloud Map – 其中一個憑證 SANs 必須符合使用 格式 AWS Cloud Map 的服務探索設定中提供的值*service-name.namespace-name*。對於具有 serviceName AWS Cloud Map 的服務探索設定*mesh-endpoint*和 namespaceName 的應用程式*apps.local*，您可以建立與名稱 相符的憑證*mesh-endpoint.apps.local*，或使用萬用字元建立憑證 *.*apps.local*.

對於這兩種探索機制，如果憑證 SANs都不符合 DNS 服務探索設定，Envoys 之間的連線會失敗，並顯示下列錯誤訊息，如從用戶端 Envoy 所見。

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

TLS 身分驗證憑證

使用 TLS 身分驗證時，App Mesh 支援憑證的多個來源。

AWS Private CA

憑證必須存放在與將使用憑證的網格端點相同的區域和 AWS 帳戶中的 ACM 中。CA 的憑證不需要位於相同的 AWS 帳戶中，但它仍然需要位於與網格端點相同的 區域。如果您沒有 AWS 私有 CA，則必須先建立一個，才能從中請求憑證。如需 AWS Private CA 使用 ACM 從現有 請求憑證的詳細資訊，請參閱請求私有憑證。憑證不能是公有憑證。

您用於 TLS 用戶端政策的私有 CAs 必須是根使用者 CAs。

若要設定具有憑證和 CAs虛擬節點 AWS Private CA，您用來呼叫 App Mesh 的主體 （例如使用者或角色） 必須具有下列 IAM 許可：

- 對於您新增至接聽程式 TLS 組態的任何憑證，委託人必須擁有 `acm:DescribeCertificate`許可。
- 對於在 TLS 用戶端政策上設定的任何 CAs，委託人必須擁有 `acm-pca:DescribeCertificateAuthority`許可。

> **Important**
>
> 與其他帳戶共用 CAs可能會將這些帳戶意外的權限授予 CA。我們建議使用資源型政策，針對不需要從 CA 發行憑證`acm-pca:GetCertificateAuthorityCertificate`的帳戶，限制僅對 `acm-pca:DescribeCertificateAuthority`和 的存取。

您可以將這些許可新增至連接至委託人的現有 IAM 政策，或建立新的委託人和政策，並將政策連接至委託人。如需詳細資訊，請參閱編輯 IAM 政策、建立 IAM 政策和新增 IAM 身分許可。

> **Note**
>
> 您需要為每個 的操作支付每月費用， AWS Private CA 直到您將其刪除為止。您還需要為每月發行的私有憑證和匯出的私有憑證付費。如需詳細資訊，請參閱AWS Certificate Manager 定價。

當您為網格端點代表的 Envoy Proxy 啟用代理授權時，您必須指派您使用的 IAM 角色下列 IAM 許可：

- 對於在虛擬節點接聽程式上設定的任何憑證，角色必須具有 `acm:ExportCertificate`許可。
- 對於在 TLS 用戶端政策上設定的任何 CAs，角色必須具有 `acm-pca:GetCertificateAuthorityCertificate`許可。

檔案系統

您可以使用 檔案系統將憑證分發給 Envoy。您可以透過讓憑證鏈和對應的私有金鑰在檔案路徑上可用來執行此操作。如此一來，即可從 Envoy 附屬代理存取這些資源。

Envoy 的秘密探索服務 (SDS)

Envoy 透過 Secrets Discovery 通訊協定，從特定端點擷取 TLS 憑證等秘密。如需此通訊協定的詳細資訊，請參閱 Envoy 的 SDS 文件。

當 SDS 做為憑證和憑證鏈的來源時，App Mesh 會將 Envoy 代理設定為使用代理本機的 Unix Domain Socket 做為 Secret Discovery Service (SDS) 端點。您可以使用 `APPMESH_SDS_SOCKET_PATH`環境變數來設定此端點的路徑。

> **Important**
>
> App Mesh Envoy 代理版本 1.15.1.0 及更新版本支援使用 Unix Domain Socket 的 Local Secrets Discovery Service。
> App Mesh 支援使用 gRPC 的 V2 SDS 通訊協定。

與 SPIFFE 執行期環境 (SPIRE) 整合

您可以使用 SDS API 的任何附屬實作，包括 SPIFFE 執行期環境 (SPIRE) 等現有工具鏈。SPIRE 旨在啟用分散式系統中多個工作負載之間的交互 TLS 身分驗證部署。它會在執行時間證明工作負載的身分。SPIRE 也會將特定工作負載、短期和自動輪換金鑰和憑證直接交付至工作負載。

您應該將 SPIRE Agent 設定為 Envoy 的 SDS 供應商。允許它為 Envoy 直接提供其提供交互 TLS 身分驗證所需的金鑰材料。在 Envoy 代理旁邊的附屬中執行 SPIRE 代理程式。代理程式會視需要負責重新產生短期金鑰和憑證。代理程式會證明 Envoy，並判斷當 Envoy 連線到 SPIRE 代理程式公開的 SDS 伺服器時，應該提供給 Envoy 哪些服務身分和 CA 憑證。

在此過程中，服務身分和 CA 憑證會輪換，並將更新串流回 Envoy。Envoy 會立即將它們套用至新的連線，而不會中斷或停機，也不會讓私有金鑰接觸檔案系統。

App Mesh 如何設定 Envoy 來交涉 TLS

App Mesh 會在決定如何在網格中設定 Envoys 之間的通訊時，同時使用用戶端和伺服器的網格端點組態。

使用用戶端政策

當用戶端政策強制執行 TLS 的使用，且用戶端政策中的其中一個連接埠符合伺服器政策的連接埠時，用戶端政策會用來設定用戶端的 TLS 驗證內容。例如，如果虛擬閘道的用戶端政策符合虛擬節點的伺服器政策，則會使用虛擬閘道用戶端政策中定義的設定，在代理之間嘗試 TLS 交涉。如果用戶端政策不符合伺服器政策的連接埠，則根據伺服器政策的 TLS 設定，代理之間的 TLS 可能會也可能不會交涉。

沒有用戶端政策

如果用戶端尚未設定用戶端政策，或用戶端政策與伺服器的連接埠不相符，App Mesh 將使用伺服器來判斷是否要從用戶端交涉 TLS，以及如何進行。例如，如果虛擬閘道尚未指定用戶端政策，且虛擬節點尚未設定 TLS 終止，則代理之間不會交涉 TLS。如果用戶端未指定相符的用戶端政策，且

伺服器已設定為 TLS 模式STRICT或 PERMISSIVE，則代理會設定為交涉 TLS。視憑證如何提供用於 TLS 終止而定，適用下列額外行為。

- ACM 受管 TLS 憑證 – 當伺服器已使用 ACM 受管憑證設定 TLS 終止時，App Mesh 會自動設定用戶端以交涉 TLS，並根據憑證鏈結所在的根使用者 CA 驗證憑證。
- 以檔案為基礎的 TLS 憑證 – 當伺服器已使用代理本機檔案系統的憑證設定 TLS 終止時，App Mesh 會自動設定用戶端來交涉 TLS，但不會驗證伺服器的憑證。

主體替代名稱

您可以選擇性地指定要信任的主體別名 (SANs清單。SANs必須為 FQDN 或 URI 格式。如果提供 SANs，Envoy 會驗證所出示憑證的主體別名是否符合此清單上的其中一個名稱。

如果您未在終止網格端點上指定 SAN，則該節點的 Envoy 代理不會在對等用戶端憑證上驗證 SAN。如果您未在原始網格端點上指定 SAN，則終止端點提供的憑證上的 SAN 必須與網格端點服務探索組態相符。

如需詳細資訊，請參閱 App Mesh TLS：憑證需求。

> **Important**
>
> 只有在 TLS 的用戶端政策設定為 時，才能使用萬用字元 SANsnot enforced。如果用戶端虛擬節點或虛擬閘道的用戶端政策設定為強制執行 TLS，則無法接受萬用字元 SAN。

驗證加密

啟用 TLS 之後，您可以查詢 Envoy 代理，以確認通訊已加密。Envoy 代理會發出 資源的統計資料，協助您了解 TLS 通訊是否正常運作。例如，Envoy 代理會記錄針對指定網格端點交涉的成功 TLS 交握次數統計資料。*my-mesh-endpoint* 使用以下命令，判斷名為 的網格端點有多少成功 TLS 交握。

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep ssl.handshake
```

在下列範例傳回的輸出中，網格端點有三個交握，因此會加密通訊。

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

Envoy 代理也會在 TLS 交涉失敗時發出統計資料。判斷網格端點是否有 TLS 錯誤。

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep -e "ssl.*\(fail\|error
\)"
```

在範例傳回的輸出中，數個統計資料沒有錯誤，因此 TLS 交涉成功。

```
listener.0.0.0.0_15000.ssl.connection_error: 0
listener.0.0.0.0_15000.ssl.fail_verify_cert_hash: 0
listener.0.0.0.0_15000.ssl.fail_verify_error: 0
listener.0.0.0.0_15000.ssl.fail_verify_no_cert: 0
listener.0.0.0.0_15000.ssl.ssl.fail_verify_san: 0
```

如需 Envoy TLS 統計資料的詳細資訊，請參閱 Envoy Listener Statistics。

憑證續約

AWS Private CA

當您使用 ACM 續約憑證時，續約的憑證會在續約完成後的 35 分鐘內自動分發到連線的代理。我們建議您使用 受管續約，在接近有效期間結束時自動續約憑證。如需詳細資訊，請參閱 AWS Certificate Manager 《使用者指南》中的 ACM Amazon 發行憑證的受管續約。

您自己的憑證

使用本機檔案系統的憑證時，Envoy 不會在憑證變更時自動重新載入憑證。您可以重新啟動或重新部署 Envoy 程序，以載入新的憑證。您也可以將較新的憑證放在不同的檔案路徑，並使用該檔案路徑更新虛擬節點或閘道組態。

設定 Amazon ECS 工作負載以搭配 使用 TLS 身分驗證 AWS App Mesh

您可以設定您的網格以使用 TLS 身分驗證。請確定憑證可供您新增至工作負載的 Envoy 代理附屬項目使用。您可以將 EBS 或 EFS 磁碟區連接到 Envoy 附屬，也可以從 AWS Secrets Manager 存放和擷取憑證。

- 如果您使用以檔案為基礎的憑證分佈，請將 EBS 或 EFS 磁碟區連接至您的 Envoy 附屬。確定憑證和私有金鑰的路徑符合其中設定的路徑 AWS App Mesh。
- 如果您使用的是 SDS 型分佈，請新增可實作 Envoy 的 SDS API 並存取憑證的附屬項目。

> **Note**
>
> Amazon ECS 不支援 SPIRE。

設定 Kubernetes 工作負載以搭配 使用 TLS 身分驗證 AWS App Mesh

您可以設定 Kubernetes 的 AWS App Mesh 控制器，為虛擬節點和虛擬閘道服務後端和接聽程式啟用 TLS 身分驗證。確定憑證可供您新增至工作負載的 Envoy 代理附屬。您可以在相互 TLS 身分驗證的逐步解說區段中查看每個分佈類型的範例。

- 如果您使用以檔案為基礎的憑證分佈，請將 EBS 或 EFS 磁碟區連接至您的 Envoy 附屬。確定憑證和私有金鑰的路徑與控制器中設定的路徑相符。或者，您可以使用掛載在檔案系統的 Kubernetes 秘密。
- 如果您使用的是 SDS 型分佈，您應該設定節點本機 SDS 供應商，以實作 Envoy 的 SDS API。Envoy 將透過 UDS 到達它。若要在 EKS AppMesh 控制器中啟用 SDS 型 mTLS 支援，請將 `enable-sds`旗標設定為 ，`true`並透過 `sds-uds-path`旗標將本機 SDS 供應商的 UDS 路徑提供給控制器。如果您使用 helm，您可以將這些設定為控制器安裝的一部分：

```
--set sds.enabled=true
```

> **Note**
>
> 如果您在 Fargate 模式下使用 Amazon Elastic Kubernetes Service (Amazon EKS)，則無法使用 SPIRE 來分發憑證。

交互 TLS 驗證

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

相互 TLS (Transport Layer Security) 身分驗證是 TLS 的選用元件，可提供雙向對等身分驗證。相互 TLS 身分驗證透過 TLS 新增一層安全性，並允許您的服務驗證正在建立連線的用戶端。

用戶端-伺服器關係中的用戶端也會在工作階段交涉過程中提供 X.509 憑證。伺服器使用此憑證來識別和驗證用戶端。此程序有助於驗證憑證是否由信任的憑證授權機構 (CA) 發行，以及憑證是否為有效的憑證。它也會使用憑證上的主體別名 (SAN) 來識別用戶端。

您可以為 支援的所有通訊協定啟用相互 TLS 身分驗證 AWS App Mesh。它們是 TCP、HTTP/1.1、HTTP/2、gRPC。

> **Note**
>
> 使用 App Mesh，您可以為來自 服務的 Envoy 代理之間的通訊設定相互 TLS 身分驗證。不過，您的應用程式與 Envoy 代理之間的通訊會遭到取消加密。

相互 TLS 身分驗證憑證

AWS App Mesh 支援兩個可能的憑證來源，以進行相互 TLS 身分驗證。TLS 用戶端政策中的用戶端憑證和接聽程式 TLS 組態中的伺服器驗證可從下列來源取得：

- 檔案系統 – 來自正在執行之 Envoy 代理的本機檔案系統的憑證。若要將憑證分發給 Envoy，您需要提供憑證鏈的檔案路徑和 App Mesh API 的私有金鑰。
- Envoy 的秘密探索服務 (SDS) – Bring-your-own的附屬工具，可實作 SDS 並允許將憑證傳送至 Envoy。其中包括 SPIFFE 執行期環境 (SPIRE)。

> **Important**
>
> App Mesh 不會存放用於相互 TLS 身分驗證的憑證或私有金鑰。相反地，Envoy 會將它們存放在記憶體中。

設定網格端點

為您的網格端點設定相互 TLS 身分驗證，例如虛擬節點或閘道。這些端點提供憑證並指定信任的機構。

若要這樣做，您需要同時為用戶端和伺服器佈建 X.509 憑證，並在 TLS 終止和 TLS 原始伺服器的驗證內容中明確定義信任的授權憑證。

網格內部的信任

伺服器端憑證是在 Virtual Node 接聽程式 (TLS 終止） 中設定，而用戶端憑證是在 Virtual Nodes 服務後端 (TLS 原始伺服器） 中設定。作為此組態的替代方案，您可以為虛擬節點的所有服務後端定義預設用戶端政策，然後，如有需要，您可以針對特定後端覆寫此政策。虛擬閘道只能使用套用到其所有後端的預設用戶端政策來設定。

您可以透過為兩個網格的虛擬閘道上的傳入流量啟用相互 TLS 身分驗證，來設定不同網格之間的信任。

在網格外信任

在 Virtual Gateway 接聽程式中指定伺服器端憑證以進行 TLS 終止。設定與虛擬閘道通訊的外部服務，以呈現用戶端憑證。憑證應衍生自伺服器端憑證在 Virtual Gateway 接聽程式上用於 TLS 起始的相同憑證授權機構 (CAs) 之一。

將服務遷移至相互 TLS 身分驗證

在 App Mesh 中將現有服務遷移至相互 TLS 身分驗證時，請遵循這些準則來維持連線。

遷移透過純文字通訊的服務

1. 在伺服器端點上啟用 TLS 組態的`PERMISSIVE`模式。此模式允許純文字流量連線到端點。
2. 在伺服器上設定相互 TLS 身分驗證，指定伺服器憑證、信任鏈，以及選用的信任 SANs。
3. 確認通訊是透過 TLS 連線進行。
4. 在用戶端上設定相互 TLS 身分驗證，指定用戶端憑證、信任鏈，以及選用的信任 SANs。
5. 在伺服器上啟用 TLS 組態的`STRICT`模式。

遷移透過 TLS 通訊的服務

1. 在用戶端上設定相互 TLS 設定，指定用戶端憑證和選用的信任 SANs。在後端伺服器請求用戶端憑證之前，用戶端憑證不會傳送至其後端。
2. 在伺服器上設定相互 TLS 設定，指定信任鏈和選用的信任 SANs。為此，您的伺服器會請求用戶端憑證。

驗證相互 TLS 身分驗證

您可以參考 Transport Layer Security：驗證加密文件，了解 Envoy 如何準確發出 TLS 相關統計資料。對於相互 TLS 身分驗證，您應該檢查下列統計資料：

- `ssl.handshake`
- `ssl.no_certificate`
- `ssl.fail_verify_no_cert`
- `ssl.fail_verify_san`

下列兩個統計資料範例一起顯示，成功終止虛擬節點的 TLS 連線，全都來自提供憑證的用戶端。

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

```
listener.0.0.0.0_15000.ssl.no_certificate: 0
```

下一個統計資料範例顯示從虛擬用戶端節點（或閘道）到後端虛擬節點的連線失敗。伺服器憑證中顯示的主題別名 (SAN) 不符合用戶端信任的任何 SANs。

```
cluster.cds_egress_my-mesh_my-backend-node_http_9080.ssl.fail_verify_san: 5
```

App Mesh 相互 TLS 身分驗證演練

- 相互 TLS 身分驗證演練：此演練說明如何使用 App Mesh CLI 建立具有相互 TLS 身分驗證的色彩應用程式。
- Amazon EKS 相互 TLS SDS 型演練：此演練說明如何透過 Amazon EKS 和 SPIFFE 執行期環境 (SPIRE) 使用相互 TLS SDS 型身分驗證。
- Amazon EKS 相互 TLS 檔案型演練：此演練說明如何搭配 Amazon EKS 和 SPIFFE 執行期環境 (SPIRE) 使用相互 TLS 檔案型身分驗證。

AWS App Mesh 如何使用 IAM

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS Identity and Access Management (IAM) 是一種 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （ 登入 ） 和授權 （ 具有許可 ） 來使用 App Mesh 資源。IAM 是 AWS 服務 您可以免費使用的 。

主題

目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 App Mesh 中所做的工作。

服務使用者 – 如果您使用 App Mesh 服務來執行您的任務，您的管理員會為您提供所需的登入資料和許可。當您使用更多 App Mesh 功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 App Mesh 中的功能，請參閱 對 AWS App Mesh 身分和存取進行故障診斷。

服務管理員 – 如果您在公司負責 App Mesh 資源，您可能擁有 App Mesh 的完整存取權。您的任務是判斷服務使用者應存取哪些 App Mesh 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來

變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何搭配 App Mesh 使用 IAM，請參閱AWS App Mesh 如何使用 IAM。

IAM 管理員 – 如果您是 IAM 管理員，建議您了解撰寫政策以管理 App Mesh 存取的詳細資訊。若要檢視您可以在 IAM 中使用的 App Mesh 身分型政策範例，請參閱 AWS App Mesh 身分型政策範例。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以身分驗證 （ 登入 AWS) 做為 AWS 帳戶根使用者、以 IAM 使用者身分驗證，或擔任 IAM 角色驗證。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分身分登入 。 AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取 時，您會間接擔任角色。

視您身分的使用者類型而定，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱AWS 登入 《 使用者指南》中的如何登入您的 AWS 帳戶 。

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的登入資料以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的適用於 API 請求的AWS Signature 第 4 版。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的 IAM 中的AWS 多重要素驗證。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取 帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是 中具有單一個人或應用程式特定許可 AWS 帳戶 的身分。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期

憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從使用者切換至 IAM 角色（主控台）。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《IAM 使用者指南》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以直接將政策連接到資源（而不是使用角色做為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，並結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要

與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

- 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。
- 服務連結角色 – 服務連結角色是連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體，並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式。

使用政策管理存取權

您可以透過建立政策並將其連接到 AWS 身分或資源 AWS 來控制 中的存取。政策是 中的物件， AWS 當與身分或資源建立關聯時， 會定義其許可。當委託人（使用者、根使用者或角色工作階段）提出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console、 AWS CLI或 API AWS 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制政策 SCPs) – SCPs是 JSON 政策，可指定 中組織或組織單位 (OU) 的最大許可 AWS Organizations。 AWS Organizations 是用於分組和集中管理您企業擁有 AWS 帳戶 之多個的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。

如需 Organizations 和 RCPs的詳細資訊，包括 AWS 服務 支援 RCPs的 清單，請參閱AWS Organizations 《 使用者指南》中的資源控制政策 RCPs)。

- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

AWS App Mesh 如何使用 IAM

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

使用 IAM 管理 App Mesh 的存取權之前，您應該了解哪些 IAM 功能可與 App Mesh 搭配使用。若要深入了解 App Mesh 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的服務。

主題

- App Mesh 身分型政策
- App Mesh 資源型政策
- 以 App Mesh 標籤為基礎的授權
- App Mesh IAM 角色

App Mesh 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。App Mesh 支援特定動作、資源和條件索引鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 JSON 政策元素參考。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

App Mesh 中的政策動作在動作之前使用下列字首：appmesh:。例如，若要授予某人使用 appmesh:ListMeshes API 操作列出帳戶中網格的許可，請在其政策中包含 appmesh:ListMeshes動作。政策陳述式必須包含 Action 或 NotAction 元素。

若要在單一陳述式中指定多個 動作，請用逗號分隔，如下所示。

```
"Action": [
      "appmesh:ListMeshes",
      "appmesh:ListVirtualNodes"
]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，如需指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "appmesh:Describe*"
```

若要查看 App Mesh 動作清單，請參閱《IAM 使用者指南》中的 定義的動作 AWS App Mesh。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

App Mesh mesh 資源具有下列 ARN。

```
arn:${Partition}:appmesh:${Region}:${Account}:mesh/${MeshName}
```

如需 ARNs 格式的詳細資訊，請參閱 Amazon Resource Names (ARNs) AWS 和服務命名空間。

例如，若要在陳述式的區域#區域中指定名為 的網格####，請使用下列 ARN。

```
arn:aws:appmesh:Region-code:111122223333:mesh/apps
```

如需指定屬於特定帳戶的所有執行個體，請使用萬用字元 (*)。

```
"Resource": "arn:aws:appmesh:Region-code:111122223333:mesh/*"
```

有些 App Mesh 動作，例如用於建立資源的動作，無法對特定資源執行。在這些情況下，您必須使用萬用字元 (*)。

```
"Resource": "*"
```

許多 App Mesh API 動作都涉及多個資源。例如， CreateRoute會建立具有虛擬節點目標的路由，因此 IAM 使用者必須具有使用路由和虛擬節點的許可。若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualRouter/serviceB/route/
*",
      "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualNode/serviceB"
]
```

若要查看 App Mesh 資源類型及其 ARNs 的清單，請參閱《IAM 使用者指南》中的由 定義的資源 AWS App Mesh。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS App Mesh定義的動作。

條件索引鍵

App Mesh 支援使用一些全域條件金鑰。若要查看 AWS 全域條件金鑰，請參閱 IAM 使用者指南中的 AWS 全域條件內容金鑰。若要查看 App Mesh 支援的全域條件金鑰清單，請參閱《IAM 使用者指南》

中的 的條件金鑰 AWS App Mesh。若要了解您可以使用哪些動作和資源搭配條件索引鍵，請參閱 動作定義依據 AWS App Mesh。

範例

若要檢視 App Mesh 身分型政策的範例，請參閱 AWS App Mesh 身分型政策範例。

App Mesh 資源型政策

App Mesh 不支援以資源為基礎的政策。不過，如果您使用 AWS Resource Access Manager (AWS RAM) 服務跨 AWS 服務共用網格，則服務會將資源型政策套用至您的網格 AWS RAM 。如需詳細資訊，請參閱授予網格的許可。

以 App Mesh 標籤為基礎的授權

您可以將標籤連接至 App Mesh 資源，或將請求中的標籤傳遞至 App Mesh。如需根據標籤控制存取，請使用 `appmesh:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。如需標記 App Mesh 資源的詳細資訊，請參閱標記 AWS 資源。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱建立具有限制標籤的應用程式網格。

App Mesh IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的實體。

搭配 App Mesh 使用臨時憑證

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫 AssumeRole 或 GetFederationToken 等 AWS STS API 操作來取得臨時安全登入資料。

App Mesh 支援使用臨時憑證。

服務連結角色

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代表您完成 動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

App Mesh 支援服務連結角色。如需建立或管理 App Mesh 服務連結角色的詳細資訊，請參閱使用 App Mesh 的服務連結角色。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

App Mesh 不支援服務角色。

AWS App Mesh 身分型政策範例

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

根據預設，IAM 使用者和角色沒有建立或修改 App Mesh 資源的許可。他們也無法使用 AWS Management Console AWS CLI或 AWS API 來執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 作業的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

主題

- 政策最佳實務
- 使用 App Mesh 主控台
- 允許使用者檢視他們自己的許可
- 建立網格
- 列出並描述所有網格
- 建立具有限制標籤的應用程式網格

政策最佳實務

以身分為基礎的政策會判斷是否有人可以在您的帳戶中建立、存取或刪除 App Mesh 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用 AWS 受管政策，將許可授予許多常見使用案例。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 使用服務動作，您也可以使用條件來授予存取服務動作的權限 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以增加安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 App Mesh 主控台

若要存取 AWS App Mesh 主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 AWS 帳戶中 App Mesh 資源的詳細資訊。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (IAM 使用者或角色) 而言，主控台就無法如預期運作。您可以將 `AWSAppMeshReadOnly` 受管政策連接至使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

對於僅對 AWS CLI 或 AWS API 進行呼叫的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台上完成此動作的許可，或使用 AWS CLI 或 AWS API 以程式設計方式完成此動作的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

建立網格

此範例示範如何建立政策，允許使用者在任何區域中為帳戶建立網格。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:CreateMesh",
            "Resource": "arn:aws:appmesh:*:123456789012:CreateMesh"
        }
    ]
}
```

列出並描述所有網格

此範例示範如何建立政策，以允許使用者唯讀存取清單並描述所有網格。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appmesh:DescribeMesh",
                "appmesh:ListMeshes"
            ],
            "Resource": "*"
        }
    ]
}
```

建立具有限制標籤的應用程式網格

您可以在 IAM 政策中使用標籤，以控制哪些標籤可以在 IAM 請求中傳遞。您可以指定可以從 IAM 使用者或角色新增、變更或移除哪些標籤鍵值對。此範例示範如何建立允許建立網格的政策，但前提是網格是使用名為 *teamName* 的標籤建立，且值為 *booksTeam*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:CreateMesh",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/teamName": "booksTeam"
                }
            }
        }
    ]
}
```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果使用者嘗試建立網格，網格必須包含名為 的標籤`teamName`和 的值`booksTeam`。如果網格不包含此標籤和值，則嘗試建立網格會失敗。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。

AWS App Mesh 的 受管政策

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常見使用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住， AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受管政策中 AWS 定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可用於現有服務時， AWS 最有可能更新受 AWS 管政策。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管政策：AWSAppMeshServiceRolePolicy

您可以將 `AWSAppMeshServiceRolePolicy` 連接到 IAM 實體。啟用存取 使用或管理 AWS 的服務和資源 AWS App Mesh。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSAppMeshServiceRolePolicy。

如需 許可詳細資訊的資訊AWSAppMeshServiceRolePolicy，請參閱 App Mesh 的服務連結角色許可。

AWS 受管政策：AWSAppMeshEnvoyAccess

您可以將 AWSAppMeshEnvoyAccess 連接到 IAM 實體。用於存取虛擬節點組態的應用程式網格 Envoy 政策。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSAppMeshEnvoyAccess。

AWS 受管政策：AWSAppMeshFullAccess

您可以將 AWSAppMeshFullAccess 連接到 IAM 實體。提供 AWS App Mesh APIs和 的完整存取權 AWS Management Console。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSAppMeshFullAccess。

AWS 受管政策：AWSAppMeshPreviewEnvoyAccess

您可以將 AWSAppMeshPreviewEnvoyAccess 連接到 IAM 實體。用於存取虛擬節點組態的應用程式網格預覽 Envoy 政策。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSAppMeshPreviewEnvoyAccess。

AWS 受管政策：AWSAppMeshPreviewServiceRolePolicy

您可以將 AWSAppMeshPreviewServiceRolePolicy 連接到 IAM 實體。啟用存取使用或管理 AWS 的服務和資源 AWS App Mesh。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSAppMeshPreviewServiceRolePolicy。

AWS 受管政策：AWSAppMeshReadOnly

您可以將 AWSAppMeshReadOnly 連接到 IAM 實體。提供 API AWS App Mesh APIs唯讀存取權 AWS Management Console。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSAppMeshReadOnly。

AWS App Mesh 受 AWS 管政策的更新

檢視自此服務開始追蹤這些變更 AWS App Mesh 以來， AWS 受管政策更新的詳細資訊。如需有關此頁面變更的自動提醒，請訂閱 AWS App Mesh 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AWSAppMeshFullAccess – 已更新政策。	更新AWSAppMeshFullAccess 以允許存取 TagResource 和 UntagResource APIs。	2024 年 4 月 24 日
AWSAppMeshServiceRolePolicy、AWSServiceRoleForAppMesh – 已更新政策。	已更新 AWSServiceRoleForAppMesh 和 AWSAppMeshServiceRolePolicy 以允許存取 AWS Cloud Map DiscoverInstancesRevision API。	2023 年 10 月 12 日

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 中的使用者和群組 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。遵循「IAM 使用者指南」的為第三方身分提供者 (聯合) 建立角色中的指示。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照「IAM 使用者指南」的為 IAM 使用者建立角色中的指示。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台) 中的指示。

使用 App Mesh 的服務連結角色

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS App Mesh use AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 App Mesh 的唯一 IAM 角色類型。服務連結角色由 App Mesh 預先定義，並包含服務 AWS 代表您呼叫其他服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 App Mesh，因為您不必手動新增必要的許可。App Mesh 定義其服務連結角色的許可，除非另有定義，否則只有 App Mesh 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除服務連結角色。這可保護您的 App Mesh 資源，因為您不會不小心移除存取資源的許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

App Mesh 的服務連結角色許可

App Mesh 使用名為 AWSServiceRoleForAppMesh 的服務連結角色 – 此角色允許 App Mesh 代表您呼叫 AWS 服務。

AWSServiceRoleForAppMesh 服務連結角色信任`appmesh.amazonaws.com`服務擔任該角色。

許可詳細資訊

- `servicediscovery:DiscoverInstances` - 允許 App Mesh 完成所有資源的動作 AWS 。
- `servicediscovery:DiscoverInstancesRevision` - 允許 App Mesh 完成所有 AWS 資源的動作。

AWSServiceRoleForAppMesh

此政策包含以下許可：

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CloudMapServiceDiscovery",
   "Effect": "Allow",
   "Action": [
    "servicediscovery:DiscoverInstances",
    "servicediscovery:DiscoverInstancesRevision"
```

```
    ],
    "Resource": "*"
   },
   {
    "Sid": "ACMCertificateVerification",
    "Effect": "Allow",
    "Action": [
     "acm:DescribeCertificate"
    ],
    "Resource": "*"
   }
  ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

為 App Mesh 建立服務連結角色

如果您在 、 AWS Management Console AWS CLI或 AWS API 中於 2019 年 6 月 5 日之後建立網格，App Mesh 會為您建立服務連結角色。若要讓 為您建立服務連結角色，您用來建立網格的 IAM 帳戶必須已連接 IAM `AWSAppMeshFullAccess` 政策，或已連接包含 `iam:CreateServiceLinkedRole` 許可的政策。若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立網格時，App Mesh 會再次為您建立服務連結角色。如果您的帳戶只包含 2019 年 6 月 5 日之前建立的網格，而且您想要將服務連結角色與這些網格搭配使用，則可以使用 IAM 主控台建立角色。

您可以使用 IAM 主控台，透過 App Mesh 使用案例建立服務連結角色。在 AWS CLI 或 AWS API 中，使用服務名稱建立`appmesh.amazonaws.com`服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

編輯 App Mesh 的服務連結角色

App Mesh 不允許您編輯 AWSServiceRoleForAppMesh 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 App Mesh 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 如果您嘗試刪除資源時，App Mesh 服務正在使用 角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除 AWSServiceRoleForAppMesh 所使用的 App Mesh 資源

1. 刪除為網格中的所有路由器定義的所有路由。
2. 刪除網格中的所有虛擬路由器。
3. 刪除網格中的所有虛擬服務。
4. 刪除網格中的所有虛擬節點。
5. 刪除網格。

完成您帳戶中所有網格的先前步驟。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、 或 AWS API 來刪除 AWSServiceRoleForAppMesh 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

App Mesh 服務連結角色支援的 區域

App Mesh 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱應用程式網格端點和配額。

Envoy Proxy 授權

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

Proxy 授權會授權在 Amazon ECS 任務內、在 Amazon EKS 上執行的 Kubernetes Pod 中或在 Amazon EC2 執行個體上執行的 Envoy 代理，從 App Mesh Envoy Management Service 讀取

一或多個網格端點的組態。對於在 04/26/2021 之前已將 Envoys 連接到其 App Mesh 端點的客戶帳戶，使用 Transport Layer Security (TLS) 的虛擬節點和虛擬閘道（有或沒有 TLS) 需要代理授權。對於想要在 04/26/2021 之後將 Envoys 連接到其 App Mesh 端點的客戶帳戶，所有 App Mesh 功能都需要代理授權。建議所有客戶帳戶啟用所有虛擬節點的代理授權，即使他們不使用 TLS，使用 IAM 來授權特定資源時仍能有安全且一致的體驗。Proxy 授權需要在 IAM 政策中指定`appmesh:StreamAggregatedResources`許可。政策必須連接到 IAM 角色，且該 IAM 角色必須連接到您託管代理的運算資源。

建立 IAM 政策

如果您希望服務網格中的所有網格端點能夠讀取所有網格端點的組態，請跳至 建立 IAM 角色。如果您想要限制個別網格端點可從中讀取組態的網格端點，則需要建立一或多個 IAM 政策。建議僅將組態可讀取的網格端點限制為在特定運算資源上執行的 Envoy 代理。建立 IAM 政策並將`appmesh:StreamAggregatedResources`許可新增至政策。下列範例政策允許在服務網格中`serviceBv2`讀取名為 `serviceBv1`和 的虛擬節點組態。無法讀取服務網格中定義的任何其他虛擬節點的組態。如需建立或編輯 IAM 政策的詳細資訊，請參閱建立 IAM 政策和編輯 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:StreamAggregatedResources",
            "Resource": [
                "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/
serviceBv1",
                "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/
serviceBv2"
            ]
        }
    ]
}
```

您可以建立多個政策，每個政策都會限制對不同網格端點的存取。

建立 IAM 角色

如果您希望服務網格中的所有網格端點能夠讀取所有網格端點的組態，則只需要建立一個 IAM 角色。如果您想要限制個別網格端點可從中讀取組態的網格端點，則需要為您在上一個步驟中建立的每個政策建立角色。完成代理執行所在的運算資源的指示。

- Amazon EKS – 如果您想要使用單一角色，則可以使用建立叢集時建立並指派給工作者節點的現有角色。若要使用多個角色，您的叢集必須符合在叢集上啟用服務帳戶的 IAM 角色中定義的要求。建立 IAM 角色，並將角色與 Kubernetes 服務帳戶建立關聯。如需詳細資訊，請參閱為您的服務帳戶建立 IAM 角色和政策，以及為您的服務帳戶指定 IAM 角色。
- Amazon ECS – 選取AWS 服務，選取 Elastic Container Service，然後在建立 IAM 角色時選取 Elastic Container Service 任務使用案例。
- Amazon EC2 – 選取AWS 服務，選取 EC2，然後在建立 IAM 角色時選取 EC2 使用案例。無論您直接在 Amazon EC2 執行個體或執行個體上執行的 Kubernetes 上託管代理，這都會適用。

如需如何建立 IAM 角色的詳細資訊，請參閱為 AWS 服務建立角色。

連接 IAM 政策

如果您希望服務網格中的所有網格端點能夠讀取所有網格端點的組態，請將AWSAppMeshEnvoyAccess受管 IAM 政策連接至您在上一個步驟中建立的 IAM 角色。如果您想要限制個別網格端點可從中讀取組態的網格端點，請將您建立的每個政策連接到您建立的每個角色。如需將自訂或受管 IAM 政策連接至 IAM 角色的詳細資訊，請參閱新增 IAM 身分許可。

連接 IAM 角色

將每個 IAM 角色連接至適當的運算資源：

- Amazon EKS – 如果您將政策連接到連接到工作者節點的角色，您可以略過此步驟。如果您建立了不同的角色，請將每個角色指派給不同的 Kubernetes 服務帳戶，並將每個服務帳戶指派給包含 Envoy 代理的個別 Kubernetes Pod 部署規格。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的為您的服務帳戶指定 IAM 角色，以及》Kubernetes 文件》中的設定 Pod 的服務帳戶。
- Amazon ECS – 將 Amazon ECS 任務角色連接至包含 Envoy 代理的任務定義。任務可以使用 EC2 或 Fargate 啟動類型部署。如需如何建立 Amazon ECS 任務角色並將其連接至任務的詳細資訊，請參閱為您的任務指定 IAM 角色。
- Amazon EC2 – IAM 角色必須連接到託管 Envoy 代理的 Amazon EC2 執行個體。如需如何將角色連接至 Amazon EC2 執行個體的詳細資訊，請參閱我已建立 IAM 角色，現在我想要將其指派給 EC2 執行個體。

確認許可

透過選取其中一個運算服務名稱，確認`appmesh:StreamAggregatedResources`許可已指派給您在其中託管代理的運算資源。

Amazon EKS

自訂政策可指派給指派給工作者節點的角色、個別 Pod 或兩者。不過，建議您僅在個別 Pod 指派政策，以便將個別 Pod 的存取限制在個別網格端點。如果政策連接至指派給工作者節點的角色，請選取 Amazon EC2 索引標籤，然後完成工作者節點執行個體在該處找到的步驟。若要判斷指派給 Kubernetes Pod 的 IAM 角色，請完成下列步驟。

1. 檢視 Kubernetes 部署的詳細資訊，其中包含您要確認 Kubernetes 服務帳戶是否已指派的 Pod。下列命令會檢視名為 *my-deployment* 之部署的詳細資訊。

   ```
   kubectl describe deployment my-deployment
   ```

   在傳回的輸出中，記下 右側的值`Service Account:`。如果`Service Account:`沒有以 開頭的行，則目前不會將自訂 Kubernetes 服務帳戶指派給部署。您需要指派一個。如需詳細資訊，請參閱 Kubernetes 文件中的設定 Pod 的服務帳戶。

2. 檢視上一個步驟中傳回之服務帳戶的詳細資訊。下列命令會檢視名為 *my-service-account* 的服務帳戶詳細資訊。

   ```
   kubectl describe serviceaccount my-service-account
   ```

   如果 Kubernetes 服務帳戶與 AWS Identity and Access Management 角色相關聯，傳回的其中一個行看起來會與下列範例類似。

   ```
   Annotations:         eks.amazonaws.com/role-arn=arn:aws:iam::123456789012:role/
   my-deployment
   ```

   在先前的範例中，`my-deployment`服務帳戶所關聯的 IAM 角色名稱。如果服務帳戶輸出不包含類似上述範例的行，則 Kubernetes 服務帳戶不會與 AWS Identity and Access Management 帳戶建立關聯，您需要將其與帳戶建立關聯。如需詳細資訊，請參閱為您的服務帳戶指定 IAM 角色。

3. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

4. 在左側導覽中，選取角色。選取您在上一個步驟中記下的 IAM 角色名稱。

5. 確認您先前建立的自訂政策或 `AWSAppMeshEnvoyAccess`受管政策已列出。如果未連接任何政策，請將 IAM 政策連接至 IAM 角色。如果您想要連接自訂 IAM 政策，但沒有自訂 IAM 政策，則需要建立具有必要許可的自訂 IAM 政策。如果已連接自訂 IAM 政策，請選取政策並確認其中包含 `"Action": "appmesh:StreamAggregatedResources"`。如果沒有，則

需要將該許可新增至您的自訂 IAM 政策。您也可以確認已列出特定網格端點的適當 Amazon Resource Name (ARN)。如果未列出 ARNs，您可以編輯政策來新增、移除或變更列出的 ARNs。如需詳細資訊，請參閱編輯 IAM 政策和 建立 IAM 政策。

6. 針對包含 Envoy 代理的每個 Kubernetes Pod 重複上述步驟。

Amazon ECS

1. 從 Amazon ECS 主控台中，選擇任務定義。
2. 選取您的 Amazon ECS 任務。
3. 在任務定義名稱頁面上，選取您的任務定義。
4. 在任務定義頁面上，選取任務角色右側的 IAM 角色名稱連結。如果未列出 IAM 角色，則您需要建立 IAM 角色，並透過更新任務定義將其連接至您的任務。
5. 在摘要頁面的許可索引標籤中，確認先前建立的自訂政策或AWSAppMeshEnvoyAccess受管政策已列出。如果未連接任何政策，請將 IAM 政策連接至 IAM 角色。如果您想要連接自訂 IAM 政策，但沒有，則需要建立自訂 IAM 政策。如果連接自訂 IAM 政策，請選取政策並確認其中包含 `"Action": "appmesh:StreamAggregatedResources"`。如果沒有，則需要將該許可新增至您的自訂 IAM 政策。您也可以確認已列出特定網格端點的適當 Amazon Resource Name (ARN)。如果未列出 ARNs，您可以編輯政策來新增、移除或變更列出的 ARNs。如需詳細資訊，請參閱編輯 IAM 政策和 建立 IAM 政策。
6. 針對包含 Envoy 代理的每個任務定義重複上述步驟。

Amazon EC2

1. 在 Amazon EC2 主控台中，選取左側導覽中的執行個體。
2. 選取託管 Envoy 代理的其中一個執行個體。
3. 在描述索引標籤中，選取 IAM 角色右側的 IAM 角色名稱連結。如果未列出 IAM 角色，則需要建立 IAM 角色。
4. 在摘要頁面的許可索引標籤中，確認先前建立的自訂政策或AWSAppMeshEnvoyAccess受管政策已列出。如果未連接任何政策，請將 IAM 政策連接至 IAM 角色。如果您想要連接自訂 IAM 政策，但沒有，則需要建立自訂 IAM 政策。如果連接自訂 IAM 政策，請選取政策並確認其中包含 `"Action": "appmesh:StreamAggregatedResources"`。如果沒有，則需要將該許可新增至您的自訂 IAM 政策。您也可以確認已列出特定網格端點的適當 Amazon Resource Name (ARN)。如果未列出 ARNs，您可以編輯政策來新增、移除或變更列出的 ARNs。如需詳細資訊，請參閱編輯 IAM 政策和 建立 IAM 政策。

5. 為您託管 Envoy 代理的每個執行個體重複上述步驟。

對 AWS App Mesh 身分和存取進行故障診斷

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

使用下列資訊來協助您診斷和修正使用 App Mesh 和 IAM 時可能遇到的常見問題。

主題

- 我無權在 App Mesh 中執行動作
- 我想要允許 AWS 帳戶外的人員存取我的 App Mesh 資源

我無權在 App Mesh 中執行動作

如果 AWS Management Console 告訴您未獲授權執行 動作，則必須聯絡管理員尋求協助。您的管理員是為您提供簽署憑證的人員。

當 `mateojackson` IAM 使用者嘗試使用主控台在名為 my-mesh 的網格中建立名為 *`my-virtual-node`* 的虛擬節點，但沒有 `appmesh:CreateVirtualNode` 許可時，會發生下列錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: appmesh:CreateVirtualNode on resource: arn:aws:appmesh:us-
east-1:123456789012:mesh/my-mesh/virtualNode/my-virtual-node
```

在此情況下，Mateo 會要求管理員更新其政策，以允許他使用 `appmesh:CreateVirtualNode`動作建立虛擬節點。

> **Note**
>
> 由於虛擬節點是在網格內建立，Mateo 的帳戶也需要 `appmesh:DescribeMesh`和 `appmesh:ListMeshes`動作，才能在主控台中建立虛擬節點。

我想要允許 AWS 帳戶外的人員存取我的 App Mesh 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 App Mesh 是否支援這些功能，請參閱 AWS App Mesh 如何使用 IAM。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個資源中提供存取權給 IAM 使用者。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的提供存取權給第三方 AWS 帳戶 擁有。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

使用 記錄 AWS App Mesh API 呼叫 AWS CloudTrail

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS App Mesh 已與 整合AWS CloudTrail，此服務提供使用者、角色或 所採取動作的記錄 AWS 服務。CloudTrail 會將 App Mesh 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 App Mesh 主控台的呼叫，以及對 App Mesh API 操作的程式碼呼叫。使用 CloudTrail 收集的資訊，您可以判斷對 App Mesh 提出的請求、提出請求的 IP 地址、提出請求的時間，以及其他詳細資訊。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM Identity Center 使用者提出。

- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務服務提出。

當您建立帳戶 AWS 帳戶 時CloudTrail 會在 中處於作用中狀態，而且您會自動存取 CloudTrail 事件歷史記錄。CloudTrail 事件歷史記錄為 AWS 區域中過去 90 天記錄的管理事件，提供可檢視、可搜尋、可下載且不可變的記錄。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的使用 CloudTrail 事件歷史記錄。檢視事件歷史記錄不會產生 CloudTrail 費用。

如需 AWS 帳戶 過去 90 天內持續記錄的事件，請建立追蹤或 CloudTrail Lake 事件資料存放區。

CloudTrail 追蹤

線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。使用 建立的所有線索 AWS Management Console 都是多區域。您可以使用 AWS CLI建立單一或多區域追蹤。建議您建立多區域追蹤，因為您擷取 AWS 區域 帳戶中所有 的活動。如果您建立單一區域追蹤，您只能檢視追蹤 AWS 區域中記錄的事件。如需追蹤的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的為您的 AWS 帳戶建立追蹤和為組織建立追蹤。

您可以透過建立追蹤，免費將持續管理事件的一個複本從 CloudTrail 傳遞至您的 Amazon S3 儲存貯體，但這樣做會產生 Amazon S3 儲存費用。如需 CloudTrail 定價的詳細資訊，請參閱 AWS CloudTrail 定價。如需 Amazon S3 定價的相關資訊，請參閱 Amazon S3 定價。

CloudTrail Lake 事件資料存放區

CloudTrail Lake 讓您能夠對事件執行 SQL 型查詢。CloudTrail Lake 會將分列式 JSON 格式的現有事件轉換為 Apache ORC 格式。ORC 是一種單欄式儲存格式，針對快速擷取資料進行了最佳化。系統會將事件彙總到事件資料存放區中，事件資料存放區是事件的不可變集合，其依據為您透過套用進階事件選取器選取的條件。套用於事件資料存放區的選取器控制哪些事件持續存在並可供您查詢。如需 CloudTrail Lake 的詳細資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 AWS CloudTrail Lake。

CloudTrail Lake 事件資料存放區和查詢會產生費用。建立事件資料存放區時，您可以選擇要用於事件資料存放區的定價選項。此定價選項將決定擷取和儲存事件的成本，以及事件資料存放區的預設和最長保留期。如需 CloudTrail 定價的詳細資訊，請參閱 AWS CloudTrail 定價。

CloudTrail 中的應用程式網格管理事件

管理事件提供在 資源上執行的管理操作的相關資訊 AWS 帳戶。這些也稱為控制平面操作。根據預設，CloudTrail 記錄管理事件。

AWS App Mesh 會將所有 App Mesh 控制平面操作記錄為管理事件。如需 App Mesh 記錄到 CloudTrail 的 AWS App Mesh 控制平面操作清單，請參閱 AWS App Mesh API 參考。

App Mesh 事件範例

一個事件代表任何來源提出的單一請求，並包含請求 API 操作的相關資訊、操作的日期和時間、請求參數等。CloudTrail 日誌檔案不是公有 API 呼叫的已排序堆疊追蹤，因此事件不會以任何特定順序顯示。

以下範例顯示的是展示 `StreamAggregatedResources` 動作的 CloudTrail 日誌項目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:d060be4ac3244e05aca4e067bfe241f8",
        "arn": "arn:aws:sts::123456789012:assumed-role/Application-TaskIamRole-
C20GBLBRLBXE/d060be4ac3244e05aca4e067bfe241f8",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "invokedBy": "appmesh.amazonaws.com"
    },
    "eventTime": "2021-06-09T23:09:46Z",
    "eventSource": "appmesh.amazonaws.com",
    "eventName": "StreamAggregatedResources",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "appmesh.amazonaws.com",
    "userAgent": "appmesh.amazonaws.com",
    "eventID": "e3c6f4ce-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "connectionId": "e3c6f4ce-EXAMPLE",
        "nodeArn": "arn:aws:appmesh:us-west-2:123456789012:mesh/CloudTrail-Test/
virtualNode/cloudtrail-test-vn",
        "eventStatus": "ConnectionEstablished",
        "failureReason": ""
    },
    "eventCategory": "Management"
}
```

如需有關 CloudTrail 記錄內容的資訊，請參閱《AWS CloudTrail 使用者指南》中的 CloudTrail record contents。

中的資料保護 AWS App Mesh

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS 共同責任模型適用於 中的資料保護 AWS App Mesh。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 App Mesh 或其他 AWS 服務 使用主控台、API AWS CLI或 AWS SDKs時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

資料加密

您的資料在使用 App Mesh 時會加密。

靜態加密

根據預設，您建立的 App Mesh 組態會靜態加密。

傳輸中加密

App Mesh 服務端點使用 HTTPS 通訊協定。Envoy 代理與 App Mesh Envoy Management Service 之間的所有通訊都會加密。如果您需要針對 Envoy 代理與 App Mesh Envoy Management Service 之間的通訊進行符合 FIPS 的加密，則可以使用 Envoy 代理容器映像的 FIPS 變體。如需詳細資訊，請參閱Envoy 影像。

虛擬節點內容器之間的通訊不會加密，但此流量不會離開網路命名空間。

的合規驗證 AWS App Mesh

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

若要了解 是否 AWS 服務 在特定合規計劃的範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱 AWS Compliance Programs。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱在 中下載報告 AWS Artifact。

使用 時的合規責任 AWS 服務 取決於資料的敏感度、您公司的合規目標，以及適用的法律和法規。AWS 提供下列資源以協助合規：

- 安全合規與治理 - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- HIPAA 合格服務參考 – 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- AWS 合規資源 – 此工作手冊和指南的集合可能適用於您的產業和位置。

- AWS 客戶合規指南 – 透過合規的角度了解共同責任模型。本指南摘要說明保護 的最佳實務， AWS 服務 並將指南映射到跨多個架構的安全控制 （ 包括國家標準和技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO))。
- AWS Config 開發人員指南中的使用規則評估資源 – AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub – 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「Security Hub 控制參考」。
- Amazon GuardDuty – 透過監控您的環境是否有可疑和惡意活動，藉此 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- AWS Audit Manager – 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，以及符合法規和產業標準的方式。

中的基礎設施安全性 AWS App Mesh

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

作為受管服務， AWS App Mesh 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱 Security Pillar AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 App Mesh。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可以設定 App Mesh 以使用界面 VPC 端點，藉此改善 VPC 的安全狀態。如需詳細資訊，請參閱App Mesh 介面 VPC 端點 (AWS PrivateLink)。

App Mesh 介面 VPC 端點 (AWS PrivateLink)

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

您可以設定 App Mesh 以使用界面 VPC 端點，藉此改善 Amazon VPC 的安全性狀態。介面端點採用 AWS PrivateLink，這項技術可讓您使用私有 IP 地址來私下存取 App Mesh APIs。PrivateLink 會將 Amazon VPC 和 App Mesh 之間的所有網路流量限制為 Amazon 網路。

您不需要 (但建議) 設定 PrivateLink。如需 PrivateLink 和介面 VPC 端點的詳細資訊，請參閱透過 AWS PrivateLink 存取服務。

App Mesh 介面 VPC 端點的考量事項

設定 App Mesh 的介面 VPC 端點之前，請注意下列考量事項：

- 如果您的 Amazon VPC 沒有網際網路閘道，且您的任務使用`awslogs`日誌驅動程式將日誌資訊傳送至 CloudWatch Logs，則必須為 CloudWatch Logs 建立介面 VPC 端點。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用 CloudWatch Events 搭配介面 VPC 端點。
- VPC 端點不支援 AWS 跨區域請求。請確定您在計劃對 App Mesh 發出 API 呼叫的相同區域中建立端點。
- 透過 Amazon Route 53，VPC 端點僅支援 Amazon 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組必須允許連接埠 443 上來自 Amazon VPC 私有子網路的傳入連線。

> **Note**
>
> Envoy 連線不支援透過將端點政策連接至 VPC 端點 （例如，使用服務名稱 `com.amazonaws.`*`Region`*`.appmesh-envoy-management`) 來控制對 App Mesh 的存取。

如需其他考量和限制，請參閱介面端點可用區域考量和介面端點屬性和限制。

建立 App Mesh 的介面 VPC 端點

若要建立 App Mesh 服務的介面 VPC 端點，請使用 Amazon VPC 使用者指南中的建立介面端點程序。`com.amazonaws.Region.appmesh-envoy-management` 為您的 Envoy 代理指定 服務名稱，以連線至 App Mesh 的公有 Envoy 管理服務，並為網格操作`com.amazonaws.Region.appmesh`指定 。

> **Note**
>
> *##*代表 App Mesh 支援的 AWS 區域的區域識別符，例如`us-east-2`美國東部（俄亥俄）區域。

雖然您可以在支援 App Mesh 的任何區域中定義 App Mesh 的介面 VPC 端點，但您可能無法為每個區域中的所有可用區域定義端點。若要了解區域中介面 VPC 端點支援哪些可用區域，請使用 describe-vpc-endpoint-services 命令或使用 AWS Management Console。例如，下列命令會傳回可用區域，您可以在美國東部（俄亥俄）區域內部署 App Mesh 界面 VPC 端點：

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.us-east-2.appmesh-envoy-management`].AvailabilityZones[]'
```

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.us-east-2.appmesh`].AvailabilityZones[]'
```

中的彈性 AWS App Mesh

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。 AWS 區域提供多個實體分隔和隔離的可用區域，這些區域與低延遲、高輸送量和高備援聯網連接。透過可用區域，您所設計與操作的應用程式

和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

App Mesh 會在多個可用區域執行其控制平面執行個體，以確保高可用性。App Mesh 會自動偵測並取代運作狀態不佳的控制平面執行個體，並提供自動化版本升級和修補。

中的災難復原 AWS App Mesh

App Mesh 服務可管理客戶資料的備份。您不需要採取任何動作來管理備份。備份的資料會加密。

中的組態和漏洞分析 AWS App Mesh

> **Important**
>
> 終止支援通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

App Mesh 提供受管 Envoy 代理 Docker 容器映像，您可以使用微服務部署。App Mesh 可確保使用最新的漏洞和效能修補程式修補容器映像。App Mesh 會根據 App Mesh 功能集測試新的 Envoy 代理版本，然後再提供映像給您。

您必須更新微服務，才能使用更新的容器映像版本。以下是映像的最新版本。

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.12.1-prod
```

App Mesh 疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本章討論疑難排解最佳實務，以及使用 App Mesh 時可能遇到的常見問題。選取下列其中一個區域，以檢視該區域的最佳實務和常見問題。

主題

App Mesh 疑難排解最佳實務

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

我們建議您遵循本主題的最佳實務，以針對使用 App Mesh 時的問題進行疑難排解。

啟用 Envoy 代理管理介面

Envoy 代理隨附管理介面，可用於探索組態和統計資料，以及執行其他管理功能，例如連線耗盡。如需詳細資訊，請參閱 Envoy 文件中的管理介面。

如果您使用受管 Envoy 影像，根據預設，管理端點會在連接埠 9901 上啟用。中的範例會將範例管理端點 URL App Mesh 設定疑難排解顯示為 `http://my-app.default.svc.cluster.local:9901/`。

> **Note**
>
> 管理端點絕不應公開至公有網際網路。此外，我們建議監控管理端點日誌，該日誌由`ENVOY_ADMIN_ACCESS_LOG_FILE`環境變數`/tmp/envoy_admin_access.log`預設設為 。

為指標卸載啟用 Envoy DogStatsD 整合

Envoy 代理可設定為卸載 OSI Layer 4 和 Layer 7 流量的統計資料，以及內部程序運作狀態的統計資料。雖然本主題說明如何使用這些統計資料，而不將指標卸載至 CloudWatch 指標和 Prometheus 等來源。將這些統計資料集中存放於所有應用程式的集中位置，可協助您更快速診斷問題和確認行為。如需詳細資訊，請參閱使用 Amazon CloudWatch 指標和 Prometheus 文件。

您可以透過設定 中定義的參數來設定 DogStatsD 指標DogStatsD 變數。如需 DogStatsD 的詳細資訊，請參閱 DogStatsD 文件。您可以在 GitHub 上的 App Mesh with Amazon ECS 基本概念中，找到指標卸載至 AWS CloudWatch 指標的示範。

啟用存取日誌

建議您在 虛擬節點和 上啟用存取日誌虛擬閘道，以探索應用程式之間傳輸流量的詳細資訊。如需詳細資訊，請參閱 Envoy 文件中的存取記錄。日誌提供有關 OSI Layer 4 和 Layer 7 流量行為的詳細資訊。當您使用 Envoy 的預設格式時，您可以使用下列剖析陳述式，使用 CloudWatch Logs Insights 分析存取日誌。

```
parse @message "[*] \"* * *\" * * * * * * * * * * *" as StartTime, Method, Path,
 Protocol, ResponseCode, ResponseFlags, BytesReceived, BytesSent, DurationMillis,
 UpstreamServiceTimeMillis, ForwardedFor, UserAgent, RequestId, Authority, UpstreamHost
```

在生產前環境中啟用 Envoy 偵錯記錄

我們建議在生產debug前環境中將 Envoy 代理的日誌層級設定為 。偵錯日誌可協助您識別問題，然後再將相關聯的 App Mesh 組態轉換為生產環境。

如果您使用的是 Envoy 映像，您可以透過ENVOY_LOG_LEVEL環境變數將日誌層級設定為 debug 。

> **Note**
>
> 我們不建議在生產環境中使用 debug關卡。將層級設定為 可debug增加記錄，並可能影響效能和卸載至 CloudWatch Logs 等解決方案的日誌整體成本。

當您使用 Envoy 的預設格式時，您可以使用下列剖析陳述式，使用 CloudWatch Logs Insights 分析程序日誌：

```
parse @message "[*][*][*][*] [*] *" as Time, Thread, Level, Name, Source, Message
```

使用 App Mesh 控制平面監控 Envoy Proxy Connectivity

我們建議您監控 Envoy 指標，control_plane.connected_state以確保 Envoy 代理與 App Mesh 控制平面通訊，以擷取動態組態資源。如需詳細資訊，請參閱 Management Server。

App Mesh 設定疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題詳細說明您在 App Mesh 設定時可能遇到的常見問題。

無法提取 Envoy 容器映像

徵狀

您在 Amazon ECS 任務中收到下列錯誤訊息。以下訊息中的 Amazon ECR *## ID* 和##可能不同，具體取決於您提取容器映像的 Amazon ECR 儲存庫。

```
CannotPullContainerError: Error response from daemon: pull access denied
 for 840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy, repository does
 not exist or may require 'docker login'
```

Resolution

此錯誤表示正在使用的任務執行角色沒有與 Amazon ECR 通訊的許可，而且無法從儲存庫提取 Envoy 容器映像。指派給 Amazon ECS 任務的任務執行角色需要具有下列陳述式的 IAM 政策：

```
{
  "Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
  ],
  "Resource": "arn:aws:ecr:us-west-2:111122223333:repository/aws-appmesh-envoy",
  "Effect": "Allow"
},
{
  "Action": "ecr:GetAuthorizationToken",
  "Resource": "*",
  "Effect": "Allow"
}
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法連線至 App Mesh Envoy 管理服務

徵狀

您的 Envoy 代理無法連線至 App Mesh Envoy 管理服務。您看到：

- 連線遭拒的錯誤
- 連線逾時
- 解決 App Mesh Envoy 管理服務端點的錯誤
- gRPC 錯誤

Resolution

請確定您的 Envoy 代理可存取網際網路或私有 VPC 端點，且您的安全群組允許連接埠 443 上的傳出流量。App Mesh 的公有 Envoy 管理服務端點遵循完整網域名稱 (FQDN) 格式。

```
# App Mesh Production Endpoint
appmesh-envoy-management.Region-code.amazonaws.com

# App Mesh Preview Endpoint
appmesh-preview-envoy-management.Region-code.amazonaws.com
```

您可以使用以下命令來偵錯 EMS 的連線。這會將有效但空白的 gRPC 請求傳送至 Envoy Management Service。

```
curl -v -k -H 'Content-Type: application/grpc' -X POST https://
appmesh-envoy-management.Region-code.amazonaws.com:443/
envoy.service.discovery.v3.AggregatedDiscoveryService/StreamAggregatedResources
```

如果您收到這些訊息，則與 Envoy Management Service 的連線會正常運作。如需偵錯 gRPC 相關錯誤，請參閱 Envoy 中與 App Mesh Envoy 管理服務中斷連線的錯誤與錯誤文字。

```
grpc-status: 16
grpc-message: Missing Authentication Token
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

Envoy 已中斷與 App Mesh Envoy 管理服務的連線，並顯示錯誤文字

徵狀

您的 Envoy 代理程式無法連線至 App Mesh Envoy 管理服務並接收其組態。您的 Envoy 代理日誌包含如下所示的日誌項目。

```
gRPC config stream closed: gRPC status code, message
```

Resolution

在大多數情況下，日誌的訊息部分應指出問題。下表列出您可能看到的最常見 gRPC 狀態碼、其原因及其解析度。

gRPC 狀態碼	原因	Resolution
0	優雅中斷與 Envoy 管理服務的連線。	沒有問題。App Mesh 偶爾會中斷 Envoy 代理與此狀態碼的連線。Envoy 會重新連線並繼續接收更新。
3	找不到網格端點（虛擬節點或虛擬閘道）或其關聯資源之一。	仔細檢查您的 Envoy 組態，以確保其具有其代表的 App Mesh 資源的適當名稱。如果您的 App Mesh 資源與其他 AWS 資源整合，例如 AWS Cloud Map 命名空間或 ACM 憑證，請確定這些資源存在。
7	Envoy 代理未經授權執行動作，例如連線至 Envoy 管理服務，或擷取相關聯的資源。	請確定您建立的 IAM 政策具有適用於 App Mesh 和其他 服務的適當政策陳述式，並將該政策連接至您的 Envoy 代理用來連線至 Envoy 管理服務的 IAM 使用者或角色。
8	指定 App Mesh 資源的 Envoy 代理數目超過帳戶層級服務配額。	如需預設帳戶配額以及如何請求提高配額的資訊，App Mesh 服務配額請參閱 。
16	Envoy 代理沒有 的有效身分驗證憑證 AWS。	請確定 Envoy 具有適當的登入資料，可透過 IAM 使用者或角色連線至 AWS 服務。如果 Envoy 程序透過1024檔案描述項使用 ，則版本 v1.24和之前的 Envoy 中已知問題 #24136 無法擷取登入資料。當 Envoy 提供高流量時，就會發生這種情況。您可以在文字「A libcurl function was given a bad argument」

gRPC 狀態碼	原因	Resolution
		的偵錯層級檢查 Envoy 日誌，以確認此問題。若要緩解此問題，請升級至 Envoy 版本 `v1.25.1.0-prod` 或更新版本。

您可以使用下列查詢，透過 Amazon CloudWatch Insights 觀察 Envoy 代理的狀態碼和訊息：

```
filter @message like /gRPC config stream closed/
| parse @message "gRPC config stream closed: *, *" as StatusCode, Message
```

如果提供的錯誤訊息沒有幫助，或您的問題仍未解決，請考慮開啟 GitHub 問題。

Envoy 容器運作狀態檢查、準備程度探查或即時性探查失敗

徵狀

您的 Envoy 代理在 Amazon ECS 任務、Amazon EC2 執行個體或 Kubernetes Pod 中運作狀態檢查失敗。例如，您可以使用下列命令查詢 Envoy 管理介面，並接收 以外的狀態`LIVE`。

```
curl -s http://my-app.default.svc.cluster.local:9901/server_info | jq '.state'
```

Resolution

以下是修復步驟的清單，取決於 Envoy 代理傳回的狀態。

- `PRE_INITIALIZING` 或 `INITIALIZING` – Envoy 代理尚未接收組態，或無法從 App Mesh Envoy 管理服務連線和擷取組態。嘗試連線時，Envoy 可能會收到來自 Envoy 管理服務的錯誤。如需詳細資訊，請參閱 中的錯誤Envoy 已中斷與 App Mesh Envoy 管理服務的連線，並顯示錯誤文字。
- `DRAINING` – Envoy 代理已開始耗盡連線，以回應 Envoy 管理界面上的 `/healthcheck/fail`或 `/drain_listeners`請求。除非您即將終止 Amazon ECS 任務、Amazon EC2 執行個體或 Kubernetes Pod，否則我們不建議在管理界面上叫用這些路徑。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

從負載平衡器到網格端點的運作狀態檢查失敗

徵狀

容器運作狀態檢查或整備調查會將您的網格端點視為運作狀態良好，但從負載平衡器到網格端點的運作狀態檢查失敗。

Resolution

若要解決問題，請完成下列任務。

- 確定與網格端點相關聯的安全群組接受您為運作狀態檢查所設定連接埠的傳入流量。
- 手動請求時，請確定運作狀態檢查持續成功；例如，來自 VPC 中的堡壘主機。
- 如果您正在為虛擬節點設定運作狀態檢查，建議您在應用程式中實作運作狀態檢查端點；例如 /ping for HTTP。這可確保 Envoy 代理和您的應用程式都可以從負載平衡器路由。
- 您可以根據所需的功能，為虛擬節點使用任何彈性負載平衡器類型。如需詳細資訊，請參閱 Elastic Load Balancing 功能。
- 如果您正在為虛擬閘道設定運作狀態檢查，建議您在虛擬閘道的接聽程式連接埠上使用具有 TCP 或 TLS 運作狀態檢查的網路負載平衡器。這可確保虛擬閘道接聽程式已引導並準備好接受連線。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

虛擬閘道不接受連接埠 1024 或以下的流量

徵狀

您的虛擬閘道不接受連接埠 1024 或更少的流量，但接受連接埠號碼大於 1024 的流量。例如，您可以使用下列命令查詢 Envoy 統計資料，並接收零以外的值。

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "update_rejected"
```

您可能會在日誌中看到類似下列文字的文字，說明無法繫結至特權連接埠：

```
gRPC config for type.googleapis.com/envoy.api.v2.Listener rejected: Error adding/
updating listener(s) lds_ingress_0.0.0.0_port_<port num>: cannot bind '0.0.0.0:<port
 num>': Permission denied
```

Resolution

若要解決問題，為閘道指定的使用者需要具有 linux 功能 CAP_NET_BIND_SERVICE。如需詳細資訊，請參閱 Linux 程式師手冊中的功能、ECS 任務定義參數中的 Linux 參數，以及 Kubernetes 文件中的設定容器的功能。

> **Important**
>
> Fargate 必須使用大於 1024 的連接埠值。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

App Mesh 連線疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題詳細介紹使用 App Mesh 連線時可能遇到的常見問題。

無法解析虛擬服務的 DNS 名稱

徵狀

您的應用程式無法解析其嘗試連線之虛擬服務的 DNS 名稱。

Resolution

這是已知問題。如需詳細資訊，請參閱依任何主機名稱命名 VirtualServices 或 FQDN GitHub 問題。App Mesh 中的虛擬服務可以命名為任何項目。只要有虛擬服務名稱的 DNS A記錄，且應用程式可以解析虛擬服務名稱，Envoy 就會代理請求並路由至其適當的目的地。若要解決問題，請將 DNS A記錄新增至`10.10.10.10`虛擬服務名稱的任何非回溯 IP 地址，例如 。DNS A記錄可在下列條件下新增：

- 在 Amazon Route 53 中，如果名稱的字尾是您的私有託管區域名稱
- 在應用程式容器的 `/etc/hosts` 檔案中
- 在您管理的第三方 DNS 伺服器中

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法連線至虛擬服務後端

徵狀

您的應用程式無法建立與虛擬節點上定義為後端之虛擬服務的連線。嘗試建立連線時，連線可能會完全失敗，或者從應用程式的角度請求可能會透過HTTP 503回應碼失敗。

Resolution

如果應用程式完全無法連線（未傳回HTTP 503回應碼），請執行下列動作：

- 請確定您的運算環境已設定為使用 App Mesh。
  - 對於 Amazon ECS，請確定您已啟用適當的代理組態。如需end-to-end逐步解說，請參閱應用程式網格和 Amazon ECS 入門。
  - 對於 Kubernetes，包括 Amazon EKS，請確定您已透過 Helm 安裝最新的 App Mesh 控制器。如需詳細資訊，請參閱 Helm Hub 上的應用程式網格控制器或教學課程：設定與 Kubernetes 的應用程式網格整合。
  - 對於 Amazon EC2，請確定您已設定 Amazon EC2 執行個體來代理 App Mesh 流量。如需詳細資訊，請參閱更新服務。
- 請確定在運算服務上執行的 Envoy 容器已成功連線至 App Mesh Envoy 管理服務。您可以檢查欄位 的 Envoy 統計資料來確認。 `control_plane.connected_state`如需 的詳細資訊`control_plane.connected_state`，請參閱 疑難排解最佳實務中的監控 Envoy Proxy Connectivity。

  如果 Envoy 最初能夠建立連線，但之後中斷連線且從未重新連線，請參閱 Envoy 中斷與 App Mesh Envoy 管理服務的連線，其中包含錯誤文字，以疑難排解中斷連線的原因。

如果應用程式已連線，但請求失敗，但有HTTP 503回應碼，請嘗試下列動作：

- 確定您連線的虛擬服務存在於網格中。
- 確定虛擬服務有提供者（虛擬路由器或虛擬節點）。
- 使用 Envoy 做為 HTTP Proxy 時，如果您看到輸出流量透過 Envoy 統計資料傳入，`cluster.cds_egress_*_mesh-allow-all`而不是正確的目的地，則很可能 Envoy 未透過正確路由請求`filter_chains`。這可能是使用不合格的虛擬服務名稱的結果。我們建議您使用實際服務的服務探索名稱做為虛擬服務名稱，因為 Envoy 代理會透過其名稱與其他虛擬服務通訊。

如需詳細資訊，請參閱虛擬服務。

- 檢查 Envoy 代理日誌是否有任何下列錯誤訊息：
  - `No healthy upstream` – Envoy 代理嘗試路由至 的虛擬節點沒有任何已解析的端點，或沒有任何運作狀態良好的端點。確定目標虛擬節點具有正確的服務探索和運作狀態檢查設定。

    如果在部署或擴展後端虛擬服務期間對 服務的請求失敗，請遵循 中的指引當虛擬服務有虛擬節點提供者503時，某些請求會失敗，但 HTTP 狀態碼會失敗。
  - `No cluster match for URL` – 當請求傳送至不符合虛擬路由器提供者定義之任何路由所定義條件的虛擬服務時，最可能發生這種情況。透過確保路徑和 HTTP 請求標頭正確，確保將來自應用程式的請求傳送至支援的路由。
  - `No matching filter chain found` – 當請求傳送到無效連接埠上的虛擬服務時，最可能發生這種情況。確定來自應用程式的請求使用虛擬路由器上指定的相同連接埠。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法連線至外部服務

徵狀

您的應用程式無法連線至網格外的服務，例如 `amazon.com`。

Resolution

根據預設，App Mesh 不允許從網格內的應用程式傳出流量到網格外的任何目的地。若要啟用與外部服務的通訊，有兩個選項：

- 將網格資源上的傳出篩選條件設定為 `ALLOW_ALL`。此設定將允許網格內的任何應用程式與網格內外的任何目的地 IP 地址通訊。
- 使用虛擬服務、虛擬路由器、路由和虛擬節點建立網格中的外部服務模型。例如，若要建立外部服務的模型`example.com`，您可以使用虛擬路由器建立名為 `example.com` 的虛擬服務，並路由將所有流量傳送到具有 DNS 服務探索主機名稱 的虛擬節點`example.com`。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法連線至 MySQL 或 SMTP 伺服器

徵狀

允許傳出流量到所有目的地 (Mesh `EgressFilter type=ALLOW_ALL`) 時，例如使用虛擬節點定義的 SMTP 伺服器或 MySQL 資料庫，來自應用程式的連線會失敗。例如，以下是嘗試連線至 MySQL 伺服器的錯誤訊息。

```
ERROR 2013 (HY000): Lost connection to MySQL server at 'reading initial communication
 packet', system error: 0
```

Resolution

這是使用 App Mesh 映像 1.15.0 版或更新版本解決的已知問題。如需詳細資訊，請參閱無法使用 App Mesh GitHub 連線至 MySQL 問題。 GitHub 發生此錯誤是因為 App Mesh 在 Envoy 中設定的傳出接聽程式新增 Envoy TLS Inspector 接聽程式篩選條件。如需詳細資訊，請參閱 Envoy 文件中的 TLS Inspector。此篩選條件會檢查從用戶端傳送的第一個封包，評估連線是否使用 TLS。不過，使用 MySQL 和 SMTP，伺服器會在連線後傳送第一個封包。如需 MySQL 的詳細資訊，請參閱 MySQL 文件中的初始交握。由於伺服器傳送第一個封包，因此篩選條件的檢查會失敗。

若要根據您的 Envoy 版本解決此問題：

- 如果您的 App Mesh 映像 Envoy 版本為 1.15.0 或更新版本，請勿將 MySQL、SMTP、MSSQL 等外部服務建模為應用程式的虛擬節點後端。
- 如果您的 App Mesh 映像 Envoy 版本早於 1.15.0，`3306`請將連接埠新增至 MySQL 服務`APPMESH_EGRESS_IGNORED_PORTS`中 的值清單，並做為您用於 STMP 的連接埠。

> **Important**
>
> 雖然標準 SMTP 連接埠是 `25`、 `587`和 `465`，但您應該只將您正在使用的連接埠新增至 `APPMESH_EGRESS_IGNORED_PORTS`，而不是全部三個連接埠。

如需詳細資訊，請參閱更新 Kubernetes 的服務、更新 Amazon ECS 的服務或更新 Amazon EC2 的服務。 Amazon EC2

如果您的問題仍未解決，您可以提供使用現有 GitHub 問題時所遇到情況的詳細資訊，或聯絡 AWS Support。

無法連線至 App Mesh 中建模為 TCP 虛擬節點或虛擬路由器的服務

徵狀

您的應用程式無法連線至使用 App Mesh PortMapping 定義中 TCP 通訊協定設定的後端。

Resolution

這是已知問題。如需詳細資訊，請參閱 GitHub 上相同連接埠上的路由至多個 TCP 目的地。由於 OSI Layer 4 上提供給 Envoy 代理的資訊限制，App Mesh 目前不允許多個以 TCP 建模的後端目的地共用相同的連接埠。若要確定 TCP 流量可針對所有後端目的地適當路由，請執行下列動作：

- 確保所有目的地都使用唯一的連接埠。如果您為後端虛擬服務使用虛擬路由器供應商，則可以變更虛擬路由器連接埠，而無需變更其路由目的地虛擬節點上的連接埠。這可讓應用程式在 Envoy 代理繼續使用虛擬節點中定義的連接埠時，在虛擬路由器連接埠上開啟連線。
- 如果建模為 TCP 的目的地是 MySQL 伺服器，或伺服器在連線後傳送第一個封包的任何其他 TCP 型通訊協定，請參閱 無法連線至 MySQL 或 SMTP 伺服器。

如果您的問題仍未解決，您可以提供我們使用現有 GitHub 問題時所遇到情況的詳細資訊，或聯絡 AWS Support。

連線成功連接到未列為虛擬節點虛擬服務後端的服務

徵狀

您的應用程式可以連接和傳送流量到未指定為虛擬節點上虛擬服務後端的目的地。

Resolution

如果請求成功到達未在 App Mesh APIs 中建模的目的地，則最可能的原因是網格的傳出篩選條件類型已設定為 `ALLOW_ALL`。當傳出篩選條件設定為 時`ALLOW_ALL`，來自您應用程式的傳出請求與模型化目的地 （ 後端 ） 不相符，將會傳送至應用程式設定的目的地 IP 地址。

如果您想要不允許流量到未在網格中建模的目的地，請考慮將傳出篩選條件值設定為 `DROP_ALL`。

> **Note**
>
> 設定網格傳出篩選條件值會影響網格內的所有虛擬節點。
> 將 `egress_filter`設定為 `DROP_ALL`並啟用 TLS 不適用於未連至 AWS 網域的傳出流量。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

當虛擬服務有虛擬節點提供者503時，某些請求會失敗，但 HTTP 狀態碼會失敗

徵狀

您應用程式的部分請求無法連線到使用虛擬節點提供者而非虛擬路由器提供者的虛擬服務後端。使用虛擬服務的虛擬路由器供應商時，請求不會失敗。

Resolution

這是已知問題。如需詳細資訊，請參閱 GitHub 上虛擬服務虛擬節點提供者的重試政策。使用虛擬節點做為虛擬服務的提供者時，您無法指定虛擬服務用戶端要使用的預設重試政策。相比之下，虛擬路由器供應商允許指定重試政策，因為它們是子路由資源的屬性。

若要減少對虛擬節點供應商的請求失敗，請改用虛擬路由器供應商，並在其路由上指定重試政策。如需減少應用程式請求失敗的其他方法，請參閱 App Mesh 最佳實務。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法連線至 Amazon EFS 檔案系統

徵狀

使用 Amazon EFS 檔案系統將 Amazon ECS 任務設定為磁碟區時，任務無法以下列錯誤開始。

```
ResourceInitializationError: failed to invoke EFS utils commands to set up EFS volumes:
 stderr: mount.nfs4: Connection refused : unsuccessful EFS utils command execution;
 code: 32
```

Resolution

這是已知問題。發生此錯誤是因為 NFS 連線至 Amazon EFS 會在任務中的任何容器啟動之前發生。此流量由代理組態路由至 Envoy，目前不會執行。由於啟動順序，NFS 用戶端無法連線至 Amazon EFS 檔案系統，且任務無法啟動。若要解決此問題，2049請將連接埠新增至 Amazon ECS 任務定義的代理組態中EgressIgnoredPorts設定的值清單。如需詳細資訊，請參閱 Proxy 組態。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

連線成功服務，但傳入的請求不會出現在 Envoy 的存取日誌、追蹤或指標中

徵狀

即使您的應用程式可以連線並傳送請求到另一個應用程式，您也無法在存取日誌中看到傳入的請求，或在 Envoy 代理的追蹤資訊中看到。

Resolution

這是已知問題。如需詳細資訊，請參閱 Github 上的 iptables 規則設定問題。Envoy 代理只會攔截其對應虛擬節點正在接聽之連接埠的傳入流量。對任何其他連接埠的請求會略過 Envoy 代理，並直接連線到其背後的服務。為了讓 Envoy 代理攔截您服務的傳入流量，您需要設定虛擬節點和服務，以便在相同的連接埠上接聽。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

在容器層級設定 HTTP_PROXY/HTTPS_PROXY 環境變數無法如預期運作。

徵狀

當 HTTP_PROXY/HTTPS_PROXY 在 設定為環境變數時：

- 任務定義中的應用程式容器已啟用 App Mesh，傳送至 App Mesh 服務命名空間的請求將從 Envoy 附屬項目取得HTTP 500錯誤回應。
- 啟用 App Mesh 的任務定義中的 Envoy 容器、從 Envoy 附屬伺服器發出的請求不會經過 HTTP/HTTPS 代理伺服器，而且環境變數將無法運作。

Resolution

對於應用程式容器：

App Mesh 函數透過讓任務中的流量通過 Envoy 代理。 HTTP_PROXY/HTTPS_PROXY 組態透過設定容器流量以通過不同的外部代理來覆寫此行為。Envoy 仍會攔截流量，但不支援使用外部代理代理來代理網格流量。

如果您想要代理所有非網格流量，請將 NO_PROXY 設定為包含網格的 CIDR/命名空間、localhost 和登入資料端點，如下列範例所示。

```
NO_PROXY=localhost,127.0.0.1,169.254.169.254,169.254.170.2,10.0.0.0/16
```

對於 Envoy 容器：

Envoy 不支援一般代理。我們不建議設定這些變數。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

即使在設定路由逾時後，上游請求也會逾時。

徵狀

您已定義下列的逾時：

- 路由，但您仍然收到上游請求逾時錯誤。
- 虛擬節點接聽程式、逾時和路由的重試逾時，但您仍然收到上游請求逾時錯誤。

Resolution

若要讓超過 15 秒的高延遲請求成功完成，您需要在路由和虛擬節點接聽程式層級指定逾時。

如果您指定大於預設 15 秒的路由逾時，請確定也會為所有參與虛擬節點的接聽程式指定逾時。不過，如果您將逾時減少到低於預設值的值，您可以選擇更新虛擬節點的逾時。如需設定虛擬節點和路由時選項的詳細資訊，請參閱虛擬節點和路由。

如果您指定重試政策，您為請求逾時指定的持續時間應一律大於或等於重試逾時乘以您在重試政策中定義的重試次數上限。這可讓您要求所有重試都成功完成。如需詳細資訊，請參閱路由。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

Envoy 回應 HTTP 錯誤請求。

徵狀

對於透過 Network Load Balancer (NLB) 傳送的所有請求，Envoy 會回應 HTTP 400 錯誤請求。檢查 Envoy 日誌時，我們會看到：

- 除錯日誌：

```
dispatch error: http/1.1 protocol error: HPE_INVALID_METHOD
```

- 存取日誌：

```
"- - HTTP/1.1" 400 DPE 0 11 0 - "-" "-" "-" "-" "-"
```

Resolution

解決方法是停用 NLB 目標群組屬性上的代理通訊協定第 2 版 (PPv2)。

截至今天，使用 App Mesh 控制平面執行的虛擬閘道和虛擬節點 Envoy 不支援 PPv2。如果您在 Kubernetes 上使用 AWS 負載平衡器控制器部署 NLB，請將下列屬性設定為 來停用 PPv2`false`：

```
service.beta.kubernetes.io/aws-load-balancer-target-group-attributes:
 proxy_protocol_v2.enabled
```

如需 NLB 資源屬性的詳細資訊，請參閱AWS Load Balancer控制器註釋。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法正確設定逾時。

徵狀

即使在虛擬節點接聽程式上設定逾時，以及在通往虛擬節點後端的路由上設定逾時後，您的請求也會在 15 秒內逾時。

Resolution

請確定後端清單包含正確的虛擬服務。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

App Mesh 擴展疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題詳細介紹您在 App Mesh 擴展時可能遇到的常見問題。

虛擬節點/虛擬閘道擴展超過 50 個複本時，連線失敗且容器運作狀態檢查失敗

徵狀

當您為虛擬節點/虛擬閘道擴展 Amazon ECS 任務、Kubernetes Pod 或 Amazon EC2 執行個體等複本數量超過 50 時，Envoy 容器運作狀態檢查會開始失敗。傳送流量至虛擬節點/虛擬閘道的下游應用程式會開始看到 HTTP 狀態碼 的請求失敗503。

Resolution

App Mesh 每個虛擬節點/虛擬閘道的 Envoy 數量的預設配額為 50。當執行中的 Envoys 數量超過此配額時，新的和目前正在執行的 Envoys 無法使用 gRPC 狀態碼 8() 連線至 App Mesh 的 Envoy 管理服務RESOURCE_EXHAUSTED。您可以提高此配額。如需詳細資訊，請參閱App Mesh 服務配額。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

當虛擬服務後端水平向外擴展或在 中擴展503時，請求會失敗

徵狀

當後端虛擬服務水平向外擴展或向內擴展時，來自下游應用程式的請求會失敗並顯示HTTP 503狀態碼。

Resolution

App Mesh 建議多種方法來緩解故障案例，同時水平擴展應用程式。如需如何防止這些失敗的詳細資訊，請參閱 App Mesh 最佳實務。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

在負載增加的情況下，Envoy 容器在隔離故障時當機

徵狀

在高流量負載下，Envoy 代理會因為分割錯誤 (Linux 結束碼) 而當機139。Envoy 程序日誌包含如下所示的陳述式。

```
Caught Segmentation fault, suspect faulting address 0x0"
```

Resolution

Envoy 代理可能已違反作業系統的預設 nofile ulimit，即一次可以開啟的檔案數量限制。此違規是由於流量導致更多連線，因此會消耗額外的作業系統通訊端。若要解決此問題，請增加主機作業系統上的 ulimit nofile 值。如果您使用的是 Amazon ECS，可以透過任務定義的資源限制設定上的 Ulimit 設定來變更此限制。 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html#container_definition_limits

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

預設資源的增加不會反映在服務限制中

徵狀

增加 App Mesh 資源的預設限制後，當您查看服務限制時，不會反映新值。

Resolution

雖然目前未顯示新的限制，但客戶仍然可以執行這些限制。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

應用程式由於大量的運作狀態檢查呼叫而當機。

徵狀

啟用虛擬節點的作用中運作狀態檢查後，運作狀態檢查呼叫的數量會上升。由於對應用程式發出的運作狀態檢查呼叫量大幅增加，應用程式會當機。

Resolution

啟用作用中運作狀態檢查時，下游（用戶端）的每個 Envoy 端點會將運作狀態請求傳送至上游叢集（伺服器）的每個端點，以便做出路由決策。因此，運作狀態檢查請求的總數為 `number of client Envoys * number of server Envoys * active health check frequency`。

若要解決此問題，請修改運作狀態檢查探查的頻率，這會降低運作狀態檢查探查的總量。除了主動運作狀態檢查之外，App Mesh 允許將極端值偵測設定為被動運作狀態檢查的方法。使用異常值偵測來設定何時根據連續`5xx`回應移除特定主機。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

App Mesh 可觀測性疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題詳細介紹使用 App Mesh 可觀測性時可能遇到的常見問題。

看不到我的應用程式的 AWS X-Ray 追蹤

徵狀

App Mesh 中的應用程式不會在 X-Ray 主控台或 APIs 中顯示 X-Ray 追蹤資訊。

Resolution

若要在 App Mesh 中使用 X-Ray，您必須正確設定元件，以啟用應用程式、附屬容器和 X-Ray 服務之間的通訊。執行下列步驟以確認 X-Ray 已正確設定：

- 確定 App Mesh Virtual Node 接聽程式通訊協定未設定為 TCP。
- 請確定與應用程式一起部署的 X-Ray 容器公開 UDP 連接埠2000並以使用者 身分執行1337。如需詳細資訊，請參閱 GitHub 上的 Amazon ECS X-Ray 範例。
- 確定 Envoy 容器已啟用追蹤。如果您使用的是 App Mesh Envoy 映像，您可以將ENABLE_ENVOY_XRAY_TRACING環境變數設定為 的值1，並將XRAY_DAEMON_PORT環境變數設定為 ，以啟用 X-Ray2000。
- 如果您已使用其中一種語言特定的 SDKs 來檢測應用程式程式碼中的 X-Ray，則請依照語言的指南來確保設定正確。
- 如果先前所有項目都設定正確，請檢閱 X-Ray 容器日誌是否有錯誤，並遵循故障診斷 AWS X-Ray中的指引。如需在 App Mesh 中整合 X-Ray 的更詳細說明，請參閱將 X-Ray 與 App Mesh 整合。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

在 Amazon CloudWatch 指標中看不到我應用程式的 Envoy 指標

徵狀

您在 App Mesh 中的應用程式不會將 Envoy 代理產生的指標發射至 CloudWatch 指標。

Resolution

當您在 App Mesh 中使用 CloudWatch 指標時，您必須正確設定多個元件，以啟用 Envoy 代理、CloudWatch 代理程式附屬項目和 CloudWatch 指標服務之間的通訊。採取下列步驟，確認 Envoy Proxy 的 CloudWatch 指標已正確設定：

- 請確定您使用 App Mesh 的 CloudWatch 代理程式映像。如需詳細資訊，請參閱 GitHub 上的 App Mesh CloudWatch 代理程式。

- 請確定您已遵循平台特定的使用說明，適當地設定適用於 App Mesh 的 CloudWatch 代理程式。如需詳細資訊，請參閱 GitHub 上的 App Mesh CloudWatch 代理程式。
- 如果先前所有項目都設定正確，請檢閱 CloudWatch 代理程式容器日誌是否有錯誤，並遵循 CloudWatch 代理程式故障診斷中提供的指引。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法設定 AWS X-Ray 追蹤的自訂取樣規則

徵狀

您的應用程式正在使用 X-Ray 追蹤，但您無法設定追蹤的取樣規則。

Resolution

由於 App Mesh Envoy 目前不支援動態 X-Ray 取樣組態，因此可使用下列解決方法。

如果您的 Envoy 版本是 `1.19.1`或更新版本，您有下列選項。

- 若要只設定取樣率，請使用 Envoy 容器上的`XRAY_SAMPLING_RATE`環境變數。值應指定為介於 `0`和 `1.00`(100%) 之間的小數。如需詳細資訊，請參閱AWS X-Ray 變數。
- 若要設定 X-Ray 追蹤器的當地語系化自訂取樣規則，請使用`XRAY_SAMPLING_RULE_MANIFEST`環境變數在 Envoy 容器檔案系統中指定檔案路徑。如需詳細資訊，請參閱《 AWS X-Ray 開發人員指南》中的取樣規則。

如果您的 Envoy 版本早於 `1.19.1`，請執行下列動作。

- 使用`ENVOY_TRACING_CFG_FILE`環境變數來變更您的取樣率。如需詳細資訊，請參閱Envoy 組態變數。指定自訂追蹤組態並定義本機取樣規則。如需詳細資訊，請參閱 Envoy X-Ray 組態。
- `ENVOY_TRACING_CFG_FILE` 環境變數的自訂追蹤組態範例：

```
tracing:
   http:
     name: envoy.tracers.xray
     typedConfig:
       "@type": type.googleapis.com/envoy.config.trace.v3.XRayConfig
       segmentName: foo/bar
       segmentFields:
         origin: AWS::AppMesh::Proxy
         aws:
```

```
        app_mesh:
          mesh_name: foo
          virtual_node_name: bar
    daemonEndpoint:
          protocol: UDP
          address: 127.0.0.1
          portValue: 2000
    samplingRuleManifest:
          filename: /tmp/sampling-rules.json
```

- 如需 `samplingRuleManifest` 屬性中取樣規則資訊清單的組態詳細資訊，請參閱設定 X-Ray SDK for Go。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

App Mesh 安全性疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題詳細介紹您在 App Mesh 安全性方面可能遇到的常見問題。

無法使用 TLS 用戶端政策連線至後端虛擬服務

徵狀

將 TLS 用戶端政策新增至虛擬節點中的虛擬服務後端時，與該後端的連線會失敗。嘗試將流量傳送至後端服務時，請求會失敗，並顯示`HTTP 503`回應碼和錯誤訊息：`upstream connect error or disconnect/reset before headers. reset reason: connection failure`。

Resolution

為了判斷問題的根本原因，我們建議您使用 Envoy 代理程序日誌來協助您診斷問題。如需詳細資訊，請參閱在生產前環境中啟用 Envoy 偵錯記錄。使用下列清單來判斷連線失敗的原因：

- 透過排除 中提到的錯誤，確保後端連線成功無法連線至虛擬服務後端。

- 在 Envoy 程序日誌中，尋找下列錯誤 （在偵錯層級記錄）。

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

此錯誤是由下列一或多個原因所造成：

- 憑證並未由 TLS 用戶端政策信任套件中定義的其中一個憑證授權機構簽署。
- 憑證不再有效 （已過期）。
- 主體別名 (SAN) 不符合請求的 DNS 主機名稱。
- 請確定後端服務提供的憑證有效、由 TLS 用戶端政策信任套件中的其中一個憑證授權機構簽署，且符合 中定義的條件Transport Layer Security (TLS)。
- 如果您收到的錯誤如下，則表示請求正在繞過 Envoy 代理並直接到達應用程式。傳送流量時，Envoy 上的統計資料不會變更，表示 Envoy 不在解密流量的路徑上。在虛擬節點的代理組態中，請確定 `AppPorts`包含應用程式正在接聽的正確值。

```
upstream connect error or disconnect/reset before headers. reset reason:
 connection failure, transport failure reason: TLS error: 268435703:SSL
 routines:OPENSSL_internal:WRONG_VERSION_NUMBER
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。如果您認為您找到了安全漏洞，或對 App Mesh 的安全性有疑問，請參閱AWS 漏洞報告準則。

當應用程式產生 TLS 時，無法連線至後端虛擬服務

徵狀

從應用程式產生 TLS 工作階段時，與後端虛擬服務的連線會失敗，而不是從 Envoy 代理。

Resolution

這是已知問題。如需詳細資訊，請參閱特徵請求：下游應用程式與上游代理 GitHub 問題之間的 TLS 交涉。 GitHub 在 App Mesh 中，目前支援來自 Envoy 代理的 TLS 原始伺服器，但不支援來自應用程式的 TLS 原始伺服器。若要在 Envoy 使用 TLS 原始伺服器支援，請在應用程式中停用 TLS 原始伺服器。這可讓 Envoy 讀取傳出請求標頭，並透過 TLS 工作階段將請求轉送至適當的目的地。如需詳細資訊，請參閱Transport Layer Security (TLS)。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。如果您認為您找到了安全漏洞，或對 App Mesh 的安全性有疑問，請參閱AWS 漏洞報告準則。

無法宣告 Envoy 代理之間的連線正在使用 TLS

徵狀

您的應用程式已在虛擬節點或虛擬閘道接聽程式上啟用 TLS 終止，或在後端 TLS 用戶端政策上啟用 TLS 原始伺服器，但您無法宣告 Envoy 代理之間的連線是透過 TLS 交涉工作階段發生。

Resolution

此解析度中定義的步驟會使用 Envoy 管理介面和 Envoy 統計資料。如需設定這些項目的說明，請參閱啟用 Envoy 代理管理介面和為指標卸載啟用 Envoy DogStatsD 整合。下列統計資料範例使用 管理介面以簡化操作。

- 針對執行 TLS 終止的 Envoy 代理：
  - 請確定已使用下列命令在 Envoy 組態中引導 TLS 憑證。

    ```
    curl http://my-app.default.svc.cluster.local:9901/certs
    ```

    在傳回的輸出中，您應該會在 中至少看到一個用於 `certificates[].cert_chain` TLS 終止的憑證項目。
  - 請確定成功傳入的代理接聽程式連線數量與 SSL 交握數量以及重複使用的 SSL 工作階段數量完全相同，如下列範例命令和輸出所示。

    ```
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep downstream_cx_total
    listener.0.0.0.0_15000.downstream_cx_total: 11
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep ssl.connection_error
    listener.0.0.0.0_15000.ssl.connection_error: 1
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep ssl.handshake
    listener.0.0.0.0_15000.ssl.handshake: 9
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep ssl.session_reused
    listener.0.0.0.0_15000.ssl.session_reused: 1
    # Total CX (11) - SSL Connection Errors (1) == SSL Handshakes (9) + SSL Sessions
     Re-used (1)
    ```
- 針對執行 TLS 起始的 Envoy 代理：
  - 請確定 TLS 信任存放區已在 Envoy 組態中使用下列命令開機。

```
curl http://my-app.default.svc.cluster.local:9901/certs
```

在 TLS 起始期間，您應該會看到至少一個用於驗證後端憑證的`certificates[].ca_certs`憑證項目。

- 請確定後端叢集的成功傳出連線數量與 SSL 交握數量以及重複使用的 SSL 工作階段數量完全相同，如下列範例命令和輸出所示。

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep upstream_cx_total
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.upstream_cx_total: 11
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep ssl.connection_error
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.connection_error: 1
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep ssl.handshake
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.handshake: 9
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep ssl.session_reused
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.session_reused: 1
# Total CX (11) - SSL Connection Errors (1) == SSL Handshakes (9) + SSL Sessions Re-used (1)
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。如果您認為您找到了安全漏洞，或對 App Mesh 的安全性有疑問，請參閱AWS 漏洞報告準則。

使用 Elastic Load Balancing 對 TLS 進行故障診斷

徵狀

嘗試設定 Application Load Balancer 或 Network Load Balancer 來加密虛擬節點的流量時，連線和負載平衡器運作狀態檢查可能會失敗。

Resolution

為了判斷問題的根本原因，您需要檢查下列項目：

- 對於執行 TLS 終止的 Envoy 代理，您需要排除任何設定錯誤。請遵循 中上述提供的步驟無法使用 TLS 用戶端政策連線至後端虛擬服務。

- 對於負載平衡器，您需要查看 的組態 TargetGroup：
  - 請確定TargetGroup連接埠符合虛擬節點定義的接聽程式連接埠。
  - 對於透過 HTTP 對服務產生 TLS 連線的 Application Load Balancer，請確定TargetGroup通訊協定設定為 HTTPS。如果正在使用運作狀態檢查，請確定 HealthCheckProtocol 設定為 HTTPS。
  - 對於透過 TCP 產生 TLS 連線的 Network Load Balancer，請確定TargetGroup通訊協定設定為 TLS。如果正在使用運作狀態檢查，請確定 HealthCheckProtocol 設定為 TCP。

> **Note**
>
> TargetGroup 需要變更TargetGroup名稱的任何更新。

正確設定後，您的負載平衡器應該使用提供給 Envoy 代理的憑證，為您的服務提供安全連線。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。如果您認為您找到安全漏洞，或對 App Mesh 的安全性有疑問，請參閱AWS 漏洞報告準則。

App Mesh Kubernetes 疑難排解

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

本主題詳細介紹當您將 App Mesh 與 Kubernetes 搭配使用時可能遇到的常見問題。

在 Kubernetes 中建立的應用程式網格資源無法在 App Mesh 中找到

徵狀

您已使用 Kubernetes 自訂資源定義 (CRD) 建立 App Mesh 資源，但當您使用 AWS Management Console 或 APIs 時，您建立的資源不會出現在 App Mesh 中。

Resolution

可能的原因為 App Mesh 的 Kubernetes 控制器發生錯誤。如需詳細資訊，請參閱 GitHub 上的故障診斷。檢查控制器日誌是否有任何錯誤或警告，指出控制器無法建立任何資源。

```
kubectl logs -n appmesh-system -f \
    $(kubectl get pods -n appmesh-system -o name | grep controller)
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

注入 Envoy 附屬裝置後，Pod 未通過整備和上線檢查

徵狀

您應用程式的 Pod 之前已成功執行，但在 Envoy 附屬裝置插入 Pod 之後，準備和上線檢查開始失敗。

Resolution

確定注入 Pod 的 Envoy 容器已使用 App Mesh 的 Envoy 管理服務開機。您可以參考 中的錯誤碼來驗證任何錯誤Envoy 已中斷與 App Mesh Envoy 管理服務的連線，並顯示錯誤文字。您可以使用下列命令來檢查相關 Pod 的 Envoy 日誌。

```
kubectl logs -n appmesh-system -f \
    $(kubectl get pods -n appmesh-system -o name | grep controller) \
    | grep "gRPC config stream closed"
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

Pod 未註冊或取消註冊為 AWS Cloud Map 執行個體

徵狀

您的 Kubernetes Pod 在其 AWS Cloud Map 生命週期中並未註冊或取消註冊。Pod 可能會成功啟動，並準備好提供流量，但不會接收任何流量。當 Pod 終止時，用戶端可能仍會保留其 IP 地址，並嘗試傳送流量給它，但失敗。

Resolution

這是已知問題。如需詳細資訊，請參閱 Pod 不會在 Kubernetes 中取得具有 GitHub 問題的自動註冊/取消註冊 AWS Cloud Map。 GitHub 由於 Pod、App Mesh 虛擬節點和資源之間的關係 AWS Cloud Map ，Kubernetes 的 App Mesh 控制器可能會取消同步並遺失資源。例如，如果在終止其關聯的 Pod 之前從 Kubernetes 刪除虛擬節點資源，則可能會發生這種情況。

若要緩解此問題：

- 請確定您正在執行最新版本的 Kubernetes App Mesh 控制器。
- 請確定虛擬節點定義中的 AWS Cloud Map `namespaceName`和 `serviceName` 正確無誤。
- 刪除虛擬節點定義之前，請務必先刪除任何相關聯的 Pod。如果您需要協助識別哪些 Pod 與虛擬節點相關聯，請參閱 無法判斷 App Mesh 資源的 Pod 執行位置。
- 如果您的問題仍然存在，請執行下列命令來檢查控制器日誌是否有錯誤，這可能有助於揭露基礎問題。

```
kubectl logs -n appmesh-system \
    $(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

- 請考慮使用以下命令重新啟動您的控制器 Pod。這可能會修正同步問題。

```
kubectl delete -n appmesh-system \
    $(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法判斷 App Mesh 資源的 Pod 執行位置

徵狀

當您在 Kubernetes 叢集上執行 App Mesh 時，運算子無法判斷工作負載或 Pod 為指定的 App Mesh 資源執行的位置。

Resolution

Kubernetes Pod 資源會以與其相關聯的網格和虛擬節點標註。您可以使用下列命令查詢指定虛擬節點名稱正在執行的 Pod。

```
kubectl get pods --all-namespaces -o json | \
    jq '.items[] | { metadata } | select(.metadata.annotations."appmesh.k8s.aws/
virtualNode" == "virtual-node-name")'
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

無法判斷 Pod 執行的 App Mesh 資源

徵狀

在 Kubernetes 叢集上執行 App Mesh 時，運算子無法判斷指定 Pod 執行的 App Mesh 資源。

Resolution

Kubernetes Pod 資源會以與其相關聯的網格和虛擬節點標註。您可以使用下列命令直接查詢 Pod，以輸出網格和虛擬節點名稱。

```
kubectl get pod pod-name -n namespace -o json | \
    jq '{ "mesh": .metadata.annotations."appmesh.k8s.aws/mesh",
 "virtualNode": .metadata.annotations."appmesh.k8s.aws/virtualNode" }'
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

Client Envoys 無法在 IMDSv1 停用的情況下與 App Mesh Envoy Management Service 通訊

徵狀

當 IMDSv1 停用時，用戶端 Envoys 無法與 App Mesh 控制平面 (Envoy Management Service) 通訊。IMDSv2在 之前的 App Mesh Envoy 版本上不支援 v1.24.0.0-prod。

Resolution

若要解決此問題，您可以執行下列三項操作之一。

- 升級至IMDSv2支援 v1.24.0.0-prod的應用程式網格 Envoy 版本 或更新版本。
- 在 Envoy 正在執行的執行個體IMDSv1上重新啟用 。如需還原 的指示IMDSv1，請參閱設定執行個體中繼資料選項。
- 如果您的服務在 Amazon EKS 上執行，建議針對服務帳戶 (IRSA) 使用 IAM 角色來擷取登入資料。如需啟用 IRSA 的指示，請參閱服務帳戶的 IAM 角色。

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

啟用 App Mesh 且注入 Envoy 時，IRSA 無法在應用程式容器上運作

徵狀

在 Amazon EKS 叢集上啟用 App Mesh 時，在 Amazon EKS 的 App Mesh 控制器的協助下，Envoy 和proxyinit容器會注入應用程式 Pod。應用程式無法擔任 IRSA ，而是擔任 node role。當我們

描述 Pod 詳細資訊時，我們會看到 AWS_WEB_IDENTITY_TOKEN_FILE或 AWS_ROLE_ARN環境變數不包含在應用程式容器中。

Resolution

如果定義了 AWS_WEB_IDENTITY_TOKEN_FILE或 AWS_ROLE_ARN環境變數，Webhook 會略過 Pod。請勿提供這些變數，Webhook 會負責為您注入這些變數。

```
reservedKeys := map[string]string{
        "AWS_ROLE_ARN":                "",
        "AWS_WEB_IDENTITY_TOKEN_FILE": "",
    }
    ...
    for _, env := range container.Env {
        if _, ok := reservedKeys[env.Name]; ok {
            reservedKeysDefined = true
        }
```

如果您的問題仍未解決，請考慮開啟 GitHub 問題或聯絡 AWS Support。

App Mesh 服務配額

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

AWS App Mesh 已與 Service Quotas 整合，此服務 AWS 可讓您從中央位置檢視和管理配額。服務配額也稱為限制。如需詳細資訊，請參閱Service Quotas 使用者指南中的什麼是 Service Quotas？。

Service Quotas 可讓您輕鬆查詢所有 App Mesh 服務配額的值。

使用 檢視 App Mesh 服務配額 AWS Management Console

1. 開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從 AWS services (AWS 服務) 清單中，搜尋並選取 AWS App Mesh。

   在服務配額清單中，您可以看到服務配額名稱、套用值 （如果有的話）、 AWS 預設配額，以及配額值是否可以調整。
4. 若要檢視服務配額的其他資訊 (例如說明)，請選擇配額名稱。

若要請求提升配額，請參閱《Service Quotas 使用者指南》中的請求提升配額。

使用 檢視 App Mesh 服務配額 AWS CLI

執行下列命令。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code appmesh \
    --output table
```

若要使用 使用更多服務配額 AWS CLI，請參閱 Service Quotas AWS CLI 命令參考。

App Mesh 的文件歷史記錄

> **Important**
>
> 支援終止通知：2026 年 9 月 30 日， AWS 將停止支援 AWS App Mesh。2026 年 9 月 30 日之後，您將無法再存取 AWS App Mesh 主控台或 AWS App Mesh 資源。如需詳細資訊，請參閱此部落格文章從 遷移 AWS App Mesh 至 Amazon ECS Service Connect。

下表說明 AWS App Mesh 使用者指南的主要更新和新功能。我們也會經常更新文件，以處理您傳送給我們的意見回饋。

變更	描述	日期
已更新AWSAppMeshFullAccess 政策	更新AWSAppMeshFullAccess 以允許存取 TagResource 和 UntagResource APIs。	2024 年 4 月 24 日
已更新 CloudTrail 整合文件	已更新描述 App Mesh 與 CloudTrail 整合以記錄 API 活動的文件。	2024 年 3 月 28 日
已更新政策	已更新 AWSServiceRoleForAppMesh 和 AWSAppMeshServiceRolePolicy 以允許存取 AWS Cloud Map DiscoverInstancesRevision API。	2023 年 10 月 12 日
App Mesh 的 VPC 端點政策支援	App Mesh 現在支援 VPC 端點政策。	2023 年 5 月 11 日
App Mesh 的多個接聽程式	App Mesh 現在支援多個接聽程式。	2022 年 8 月 18 日
適用於 App Mesh 的 IPv6	App Mesh 現在支援 IPv6。	2022 年 5 月 18 日

App Mesh Envoy Management Service 的 CloudTrail 記錄支援	App Mesh 現在支援 App Mesh Envoy Management Service 的 CloudTrail 記錄支援。	2022 年 3 月 18 日
適用於 Envoy 的應用程式網格代理程式	App Mesh 現在支援 Agent for Envoy。	2022 年 2 月 25 日
App Mesh 的多個接聽程式	（僅限應用程式網格預覽頻道）您可以為應用程式網格實作多個接聽程式。	2021 年 11 月 23 日
App Mesh 的 ARM64 支援	App Mesh 現在支援 ARM64。	2021 年 11 月 19 日
App Mesh 的指標延伸	您可以實作 App Mesh 的指標延伸。	2021 年 10 月 29 日
實作傳入流量增強功能	您可以實作主機名稱和標頭比對，並重寫主機名稱和路徑。	2021 年 6 月 14 日
實作相互 TLS 身分驗證	您可以實作相互 TLS 身分驗證。	2021 年 2 月 4 日
af-south-1 中的區域啟動	App Mesh 現在可在 af-south-1 區域中使用。	2021 年 1 月 22 日
實作相互 TLS 身分驗證	（僅限應用程式網格預覽頻道）您可以實作相互 TLS 身分驗證。	2020 年 11 月 23 日
實作虛擬閘道接聽程式的連線集區	您可以實作虛擬閘道接聽程式的連線集區。	2020 年 11 月 5 日
實作虛擬節點接聽程式的連線集區和極端值偵測	您可以實作虛擬節點接聽程式的連線集區和極端值偵測。	2020 年 11 月 5 日
在 eu-south-1 中啟動區域	App Mesh 現在可在 eu-south-1 區域中使用。	2020 年 10 月 21 日

實作虛擬閘道接聽程式的連線集區	（僅限應用程式網格預覽頻道）您可以實作虛擬閘道接聽程式的連線集區。	2020 年 9 月 28 日
實作虛擬節點接聽程式的連線集區和極端值偵測	（僅限應用程式網格預覽頻道）您可以實作虛擬節點接聽程式的連線集區和極端值偵測。	2020 年 9 月 28 日
建立網格傳入的虛擬閘道和閘道路由	啟用網格外部的資源，以與網格內部的資源通訊。	2020 年 7 月 10 日
使用適用於 Kubernetes 的 App Mesh 控制器，從 Kubernetes 內建立和管理 App Mesh 資源	您可以從 Kubernetes 內部建立和管理 App Mesh 資源。控制器也會自動將 Envoy 代理和初始化容器注入到您部署的 Pod 中。	2020 年 6 月 18 日
將逾時值新增至虛擬節點接聽程式和路由	您可以將逾時值新增至虛擬節點接聽程式和路由。	2020 年 6 月 18 日
將逾時值新增至虛擬節點接聽程式	（僅限應用程式網格預覽頻道）您可以將逾時值新增至虛擬節點接聽程式。	2020 年 5 月 29 日
建立網格傳入的虛擬閘道	（僅限應用程式網格預覽頻道）啟用網格外部的資源，以與網格內部的資源通訊。	2020 年 4 月 8 日
TLS 加密	（僅限應用程式網格預覽頻道）使用來自 AWS Private Certificate Authority 或您自己的憑證授權單位的憑證，使用 TLS 加密虛擬節點之間的通訊。	2020 年 1 月 17 日

與其他 帳戶共用網格	（僅限應用程式網格預覽頻道）您可以與其他帳戶共用網格。任何帳戶建立的資源都可以與網格中的其他資源通訊。	2020 年 1 月 17 日
將逾時值新增至路由	（僅限應用程式網格預覽頻道）您可以將逾時值新增至路由。	2020 年 1 月 17 日
在 AWS Outpost 上建立 App Mesh 代理	您可以在 AWS Outpost 上建立 App Mesh Envoy 代理。	2019 年 12 月 3 日
HTTP/2 和 gRPC 支援路由、虛擬路由器和虛擬節點	您可以路由使用 HTTP/2 和 gRPC 通訊協定的流量。您也可以將這些通訊協定的接聽程式新增至虛擬節點和虛擬路由器。	2019 年 10 月 25 日
重試政策	重試政策可讓用戶端保護自己免於間歇性的網路故障或間歇性的伺服器端故障。您可以將重試邏輯新增至路由。	2019 年 9 月 10 日
TLS 加密	（僅限應用程式網格預覽頻道）使用 TLS 加密虛擬節點之間的通訊。	2019 年 9 月 6 日
HTTP 標頭型路由	根據請求中 HTTP 標頭的存在和值來路由流量。	2019 年 8 月 15 日
應用程式網格預覽頻道的可用性	App Mesh Preview Channel 是 App Mesh 服務的不同變體。預覽頻道會公開即將推出的功能，供您在開發時嘗試。當您使用預覽頻道中的功能時，您可以透過 GitHub 提供意見回饋，以塑造功能的行為方式。	2019 年 7 月 19 日

App Mesh Interface VPC 端點 (AWS PrivateLink)	透過設定 App Mesh 以使用界面 VPC 端點來改善 VPC 的安全狀態。介面端點採用 AWS PrivateLink，這項技術可讓您使用私有 IP 地址來私下存取 App Mesh APIs。PrivateLink 會將 VPC 和 App Mesh 之間的所有網路流量限制為 Amazon 網路。	2019 年 6 月 14 日
新增 AWS Cloud Map 為虛擬節點服務探索方法	您可以指定 DNS 或 AWS Cloud Map 做為虛擬節點服務探索方法。若要使用 AWS Cloud Map 進行服務探索，您的帳戶必須具有 App Mesh 服務連結角色。	2019 年 6 月 13 日
在 Kubernetes 中自動建立 App Mesh 資源	當您在 Kubernetes 中建立資源時，建立 App Mesh 資源並將 App Mesh 附屬容器映像自動新增至 Kubernetes 部署。	2019 年 6 月 11 日
應用程式網格一般可用性	App Mesh 服務現已正式推出供生產使用。	2019 年 3 月 27 日
App Mesh API 更新	App Mesh APIs 已更新，以改善可用性。如需詳細資訊，請參閱 【BUG】 路由至具有不相符連接埠黑洞的目標虛擬節點。	2019 年 3 月 7 日
App Mesh 初始版本	服務公開預覽的初始文件	2018 年 11 月 28 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。