使用者指南

AWS App Mesh

AWS App Mesh: 使用者指南

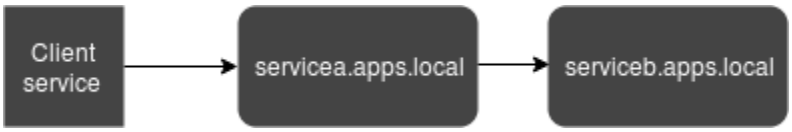
Table of Contents

什麼是 AWS App Mesh？

AWS App Mesh是一種服務網格，可以輕鬆地監控和控制服務。服務網格是一個專門用於處理通信的基礎架構層， service-to-service 通常是通過與應用程序代碼一起部署的輕量級網絡代理數組。App Mesh 標準化服務通訊方式，為您提供可 end-to-end 見性，並有助於確保 App Mesh 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。

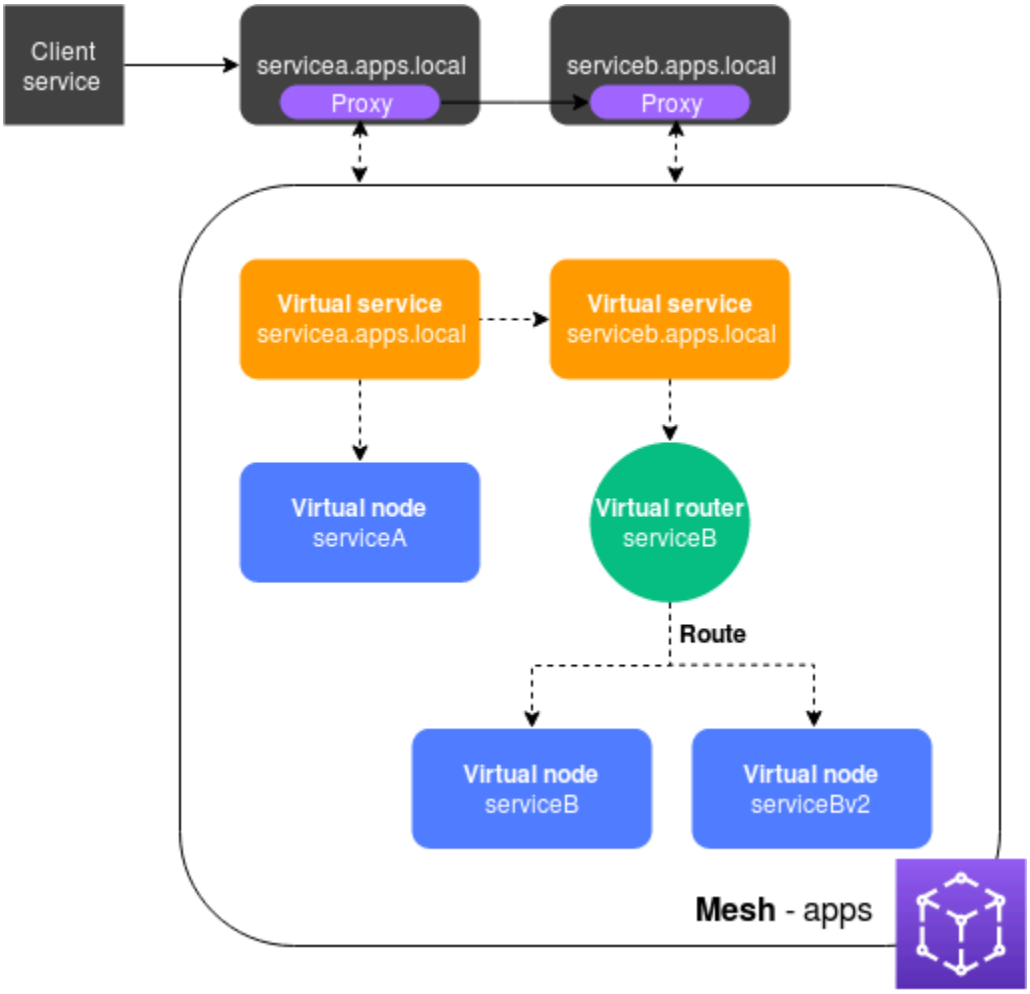
將 App Mesh 加入範例應用程式

請考慮下列不使用 App Mesh 的簡單範例應用程式。這兩種服務可以在AWS Fargate Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Compute Cloud (Amazon EC2) 執行個體 (Amazon EC2) 執行個體 (Amazon ECS) 執行個體 (Amazon ECS)、Amazon ECS、Amazon EC2 執行個體。



在此圖例中`serviceA`，`serviceB`可透過`apps.local`命名空間探索和。例如，假設您決定部署新版本的`serviceb.apps.local` named`servicebv2.apps.local`. 接下來，您想要將流量的百分比定`servicea.apps.local`向到，`serviceb.apps.local`並將百分比導向到`servicebv2.apps.local`。當您確定表現`servicebv2`良好時，您想要將 100% 的流量發送給它。

App Mesh 可以幫助您完成此操作，而無需更改任何應用程序代碼或註冊服務名稱。如果您將 App Mesh 與此範例應用程式搭配使用，則您的網格看起來可能如下圖所示。



在此配置中，服務不再直接相互通信。相反，他們通過代理相互通信。隨`servicea.apps.local`服務一起部署的 Proxy 會讀取 App Mesh 組態，並將流量傳送至`serviceb.apps.local`或`servicebv2.apps.local`根據設定傳送流量。

App Mesh

App Mesh 由下列元件組成，如上述範例所示：

- 服務網格 — 服務網格是服務之間網路流量的邏輯邊界。在此範例中，網格已命名`apps`，並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提供的實際服務的抽象化。在圖例中，兩個虛擬服務代表兩個實際的服務。虛擬服務的名稱是實際服務的可發現名稱。當虛擬服務和實際服務具有相同的名稱時，多個服務可以使用實作 App Mesh 之前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 — 虛擬節點扮演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，您將至少有一個虛擬節點。在圖例中，`servicea.apps.local`虛擬服務會取得名為之虛擬節點的組態資訊`serviceA`。`serviceA`虛擬節點設定為服務探索的`servicea.apps.local`名稱。`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內一或多個虛擬服務的流量。路由與虛擬路由器相關聯。該路由用於匹配虛擬路由器的請求，並將流量分配到其關聯的虛擬節點。在上圖中，`serviceB`虛擬路由器具有將一定百分比的流量導向`serviceB`虛擬節點的路由，並將流量的百分比導向`serviceBv2`虛擬節點。您可以設定路由至特定虛擬節點的流量百分比，並隨時間變更。您可以根據條件 (例如 HTTP 標頭、URL 路徑或 GrPC 服務和方法名稱) 路由流量。您可以設定重試原則，以便在回應中發生錯誤時重試連線。例如，在圖例中，路由的重試原則可以指定如果`serviceb.apps.local`傳回特定類型的錯誤，`serviceb.apps.local`則重試連線會重試五次，在重試嘗試之間有十秒的時間。如需詳細資訊，請參閱 虛擬路由器 及 路由。
- Proxy — 您可以將服務設定為在建立網格及其資源之後使用 Proxy。代理伺服器會讀取 App Mesh 設定並適當地導向流量。在圖例中，從`servicea.apps.local`到的所有通訊都`serviceb.apps.local`會透過與每個服務一起部署的 Proxy 進行。這些服務會使用他們在引入 App Mesh 之前使用的相同服務探索名稱彼此通訊。由於 Proxy 會讀取 App Mesh 組態，因此您可以控制兩個服務之間的通訊方式。當您想要變更 App Mesh 設定時，您不需要變更或重新部署服務本身或 Proxy。如需詳細資訊，請參閱 特使形象。

如何開始

要使用應用程序網格，您必須在亞馬遜 ECSAWS Fargate，亞馬遜 EKS，亞馬遜 EC2 上的庫伯內特斯或 Amazon EC2 上運行的現有服務。

若要開始使 App Mesh，請參閱下列其中一種指南：

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

存取 App Mesh

您可以利用下列方式來使用 App Mesh：

AWS Management Console

主控台是可供您管理 App Mesh 資源的介面。您可以在以下位置打開 App Mesh 控制台 https://console.aws.amazon.com/appmesh/。

AWS CLI

提供許多 AWS 產品的命令，且支援在 Windows、Mac 和 Linux 上使用。若要開始使用，請參閱 AWS Command Line Interface 使用者指南。若要取得 App Mesh 的相關詳細資訊，請參閱《指AWS CLI令參考》中的 appMesh。

AWS Tools for Windows PowerShell

我們也為在 PowerShell 環境中編寫指令碼編寫指令碼的使用者提供許多AWS產品的命令。若要開始使用，請參閱《 使用者指南AWS Tools for Windows PowerShell》。如需適用於 App Mesh 之指令程式的相關資訊，請參閱 PowerShellCmdlet 參考AWS工具中的 App Mesh 格。

AWS CloudFormation

可以建立範本來描述您想要的所有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》。如需有關 App Mesh 資源類型的詳細資訊，請參閱AWS CloudFormation範本參考中的 App Mesh 資源類型參考。

AWS SDK

我們也提供開發套件，可以從各種程式語言存取 App Mesh。SDK 會自動處理以下任務：

- 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 API Mesh API Mesh API 的相關詳細資訊，請參閱 AWS App MeshAPI 參考。

開始使用應用程式網格

您可以將應用程式網格與部署到 Amazon ECS、Kubernetes (部署到自己的 Amazon EC2 執行個體或在 Amazon EKS 上執行) 的應用程式搭配使用，以及 Amazon EC2。若要開始使用應用程式網格，請選取您要與 App Mesh 搭配使用之應用程式部署到的其中一個服務。完成其中一個入門指南之後，您隨時可以啟用其他服務中的應用程式也可以使用 App Mesh。

主題

開始使用 AWS App Mesh 和 Amazon ECS

本主題可協助您使 AWS App Mesh 用在 Amazon ECS 上執行的實際服務。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，請假設您有具有下列特性的應用程式：

- 由兩個名為`serviceA`和的服務組成`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% 的流量傳送`serviceA`到`serviceB`和 25% 的流量，以`serviceBv2`驗證`serviceBv2`是否沒有錯誤，然後再將 100% 的流量傳送`serviceA`至該流量。
- 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`。一旦所有流量被發送到`serviceBv2`，你想要停止`serviceB`。
- 您不需要針對實際服務變更任何現有的應用程式程式碼或服務探索註冊，以符合先前的需求。

為了滿足您的需求，您決定使用虛擬服務、虛擬節點、虛擬路由器和路由來建立 App Mesh 服務網格。實施網格後，您將更新服務以使用 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。

必要條件

- 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？。
- Amazon ECSS 概念的現有理解。如需詳細資訊，請參閱 Amazon 彈性容器服務開發人員指南中的 Amazon ECS 是什麼。
- App Mesh 支援使用 DNS (或兩者) 註冊的 Linux 服務。 AWS Cloud Map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已命名為`serviceA`、`serviceB`，`serviceBv2`且所有服務都可透過名為的命名`apps.local`空間來探索。

  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。若要建立具有服務探索的 Amazon ECS 服務，請參閱教學課程：使用服務探索建立服務。如果您尚未執行服務，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 或更高 AWS CLI 版本或 2.0.38 或更高版本來完成以下步驟。如果使用 AWS CLI，請使用指`aws --version`令來檢查已安裝的 AWS CLI 版本。如果您沒有安裝 1.18.116 或更高版本或 2.0.38 或更高版本，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 開啟 App Mesh 主控台首次執行精靈。
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 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. 對於服務探索方法，請選擇 DNS 並輸入 **serviceb.apps.local** DNS 主機名稱。
3. 在監聽器組態下，選擇 http2 做為通訊協定，然後在連接埠中輸入**80**。
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 檔案作為輸入，使用create-virtual-node指令建立虛擬節點。

```
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。虛擬路由器會在連接埠 80 上使用 HTTP/2 通訊協定監聽流量。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。添加serviceBv2虛擬節點後，權重將在稍後的步驟中。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「監聽器組態」下，選擇 http2 做為「通訊協定」，並指**80**定連接埠
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. 對於目標組態下的虛擬節點名稱，**100**為權重選擇serviceB並輸入。
6. 在比對模型組態下，選擇一個方法。
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 檔案作為輸入，使用create-virtual-router指令建立虛擬路由器。

```
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指令建立之虛擬服務的設定。

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

檢閱您使用describe-virtual-node指令建立之虛擬節點的設定。

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

檢閱您使用describe-virtual-router指令建立的虛擬路由器的設定。

```
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。對於serviceA虛擬節點，請設定的後端serviceb.apps.local。來自serviceA虛擬節點的所有輸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。
- 建立一個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。將所有流量傳送到之後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. 在監聽器組態中，選取 http2 做為通訊協定，然後在連接埠中輸入**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**serviceA**入虛擬節點名稱。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. 對於在新後端下輸入虛擬服務名稱，輸入**serviceb.apps.local**。
10. 在 [監聽器組態] 下，選擇 http2 做為 [通訊協定]，輸入**80**連接埠，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 [權重] 的值變更serviceB為**75**。
14. 選擇添加目標，serviceBv2從下拉列表中選擇，然後將重量的值設置為**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. 輸入**servicea.apps.local**虛擬服務名稱，選取提供者的虛擬節點，為虛擬節點選取serviceA虛擬節點，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：

- 授權您與每個 Amazon ECS 任務一起部署的特使代理，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，請參閱代理授權。

- 更新您現有的每個 Amazon ECS 任務定義，以使用特使代理伺服器。

登入資料

Envoy 容器需要 AWS Identity and Access Management 憑證才能簽署傳送至 App Mesh 服務的要求。對於使用 Amazon EC2 啟動類型部署的 Amazon ECS 任務，登入資料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 設定檔登入資料的 Amazon EC2 中繼資料伺服器。若要提供登入資料，您必須將 IAM 任務角色附加到任何使用 Linux 上 Fargate 容器類型部署的任務。

如果使用 Amazon EC2 啟動類型部署任務，且對 Amazon EC2 中繼資料伺服器的存取遭到封鎖 (如任務適用於 IAM 角色的重要註釋中所述)，則任務 IAM 角色也必須附加到該任務。您指派給執行個體或工作的角色必須附加 IAM 政策，如 Proxy 授權中所述。

若要使用更新您的作業定義 AWS Management Console

下列步驟僅顯示如何更新案例的 `taskB` 任務。您還需要適當地變更值來更新 `taskBv2` 和 `taskA` 任務。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 若要存取傳統的傳統 Amazon ECS 主控台，請在左上角切換「新的 EC S 體驗」。

   > **Important**
   >
   > App Mesh 整合只能在傳統的 Amazon ECS 主控台中使用。

3. 從導覽列中選擇包含您任務定義的區域。
4. 在導覽窗格中，選擇 Task Definitions (任務定義)。
5. 在 Task Definitions (任務定義) 頁面上，選取要修訂之任務定義左側的方塊。從先決條件和先前的步驟中，您可能會有名為 `taskA`、`taskB` 和 `taskBv2` 的任務定義。選取 `taskB` 並選擇 Create new revision (建立新的修訂)。
6. 在 [建立工作定義的新修訂版本] 頁面上，進行下列變更以啟用 App Mesh 整合。

   a. 對於服務整合，若要設定 App Mesh 整合的參數，請選擇啟用 App Mesh 整合，然後執行下列動作：

      i. 針對應用程式容器名稱，請選擇要用於 App Mesh 應用程式的容器名稱。此容器必須已在任務定義中先行定義。

ii. 對於 Envoy 映像檔，請完成下列工作並輸入傳回的值。

- 除了me-south-1、、、ap-east-1、ap-southeast-3和以外的所有支援區域af-south-1。eu-south-1 il-central-1您可以將區###取代為me-south-1、、、ap-east-1、ap-southeast-3和以外的任何區域。eu-south-1 il-central-1 af-south-1

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- me-south-1 區域：

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- ap-east-1 區域：

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-
prod
```

- ap-southeast-3 區域：

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- eu-south-1 區域：

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- il-central-1 區域：

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- af-south-1 區域：

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- Public repository

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod
```

> **Important**
>
> 僅支援 v1.9.0.0 版或更新版本以搭配應用程式網格使用。

iii. 針對網格名稱，請選擇要使用的 App Mesh 用程式網格服務網格。在本主題中，已建立的網格名稱為 `apps`。

iv. 對於 Virtual node name (虛擬節點名稱)，請選擇要使用的 App Mesh 虛擬節點。例如，對於 `taskB` 任務，您可以選擇您在上一個步驟中建立的 `serviceB` 虛擬節點。

v. Virtual node port (虛擬節點連接埠) 的值會預先填入您在建立虛擬節點時指定的接聽程式連接埠。

vi. 選擇 Apply (套用)，然後選擇 Confirm (確認)。即會建立新的 Envoy 代理容器並新增至任務定義，而且也會建立支援該容器的設定。然後，Envoy 代理容器會預先填入下一個步驟的 App Mesh 代理組態設定。

b. 對於 Proxy Configuration (代理組態)，請確認所有預先填入的值。

c. 對於「網路模式」，請確定`awsvpc`已選取。若要深入了解 `awsvpc` 網路模式，請參閱使用 awsvpc 網路模式進行任務聯網。

7. 選擇建立。

8. 使用更新的任務定義更新您的服務。如需詳細資訊，請參閱更新服務。

控制台創建任務定義的 json 規範。您可以修改某些設定，但無法修改其他設定。如需詳細資訊，請展開下一節。

任務定義

代理組態

若要將 Amazon ECS 服務設定為使用應用程式網格，您的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`

這是特使代理容器的入站連接埠。將此值設為 15000。

`ProxyEgressPort`

這是特使代理容器的輸出連接埠。將此值設為 15001。

`AppPorts`

指定應用程式容器偵聽的任何輸入連接埠。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。將此值設為`169.254.170.2,169.254.169.254`，會忽略 Amazon EC2 中繼資料伺服器和 Amazon ECS 任務中繼資料端點。中繼資料端點為任務登入資料提供 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**
>
> 可忽略的輸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序容器定義中設置一個dependsOn部分，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 任務定義必須包含應用程式網狀特使容器映像。

- 除了me-south-1、、、ap-east-1、ap-southeast-3和以外的所有支援區域af-south-1。eu-south-1 il-central-1您可以將區###取代為me-south-1、、、ap-east-1、ap-southeast-3和以外的任何區域。eu-south-1 il-central-1 af-south-1

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- me-south-1 區域：

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- ap-east-1 區域：

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- ap-southeast-3 區域：

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- eu-south-1 區域：

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- il-central-1 區域：

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- af-south-1 區域：

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- Public repository

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod
```

> **Important**
>
> 僅支援 v1.9.0.0 版或更新版本以搭配應用程式網格使用。

您必須使用 App Mesh Envoy 容器映像檔，直到特使專案團隊合併支援 App Mesh 的變更為止。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 單元和至少 64 MiB 的記憶體配置給特使容器。在 Fargate 上，您將能夠設置的最低值是 1024 MiB 的內存。
> - Amazon ECS 服務的虛擬節點名稱必須設定為APPMESH_RESOURCE_ARN屬性值。此屬性需要特使映像的版本`1.15.0`或更高版本。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 容器正常啟動，然後再向 Amazon ECS 報告 Envoy 容器狀態良好且可供應用程式容器啟動。
> - 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。此屬性需要特使映像的版本`1.15.0`或更高版本。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:v1.27.3.0-
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 Amazon ECS 任務定義範例顯示如何將上述範例合併到的任務定義中。taskB提供的範例可讓您針對兩種 Amazon ECS 啟動類型建立任務 (不論是否使用 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####。

如果您按照登入資料一節中所述執行 Amazon ECS 任務，則需要在範例中新增現有的任務 IAM 角色。

> **Important**
>
> Fargate 必須使用大於 1024 的端口值。

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:v1.27.3.0-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 用於 Amazon ECS 任務定義的 JSON AWS X-Ray -Linux 容器上的 Fargate

X-Ray 可讓您收集應用程式所提供請求的相關資料，並提供可用來視覺化流量流量的工具。使用特使的 X-Ray 驅動器使特使特使能夠向 X-Ray 報告跟踪信息。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以附屬容器形式執行的 X-Ray 精靈，然後守護程式會將追蹤轉送至 X-Ray 服務。將追蹤發佈至 X-Ray 後，您可以使用 X-Ray 主控台以視覺化方式呈現維修呼叫圖表並要求追蹤詳細資料。下列 JSON 代表啟用 X-Ray 整合的工作定義。

```
{


    "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:v1.27.3.0-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 適用於 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:v1.27.3.0-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 Amazon ECS 任務定義的 JSON 與 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:v1.27.3.0-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

使用應用程式網格進行金

Canary 部署和發行版本可協助您在舊版應用程式和新部署的版本之間切換流量。它也會監控新部署版本的健全狀況。如果新版本發生任何問題，初期測試部署可以自動將流量切換回舊版本。Canary 部署使您能夠在應用程序版本之間切換流量，並具有更多控制權。

如需有關如何使用 App Mesh 為 Amazon ECS 實作金絲雀部署的詳細資訊，請參閱使用應用程式網格針對 Amazon ECS 建立針對 Amazon ECS 的金絲雀部署建立管道

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

開始使用 AWS App Mesh 和庫伯尼特

當您使用適用於 Kubernetes 的應用程式網狀控制器 AWS App Mesh 與 Kubernetes 整合時，您可以管理 App Mesh 格資源，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。您也會自動將應用程式網狀邊車容器映像新增至 Kubernetes 網繭規格。本教學課程會引導您完成 Kubernetes 適用之 App Mesh 控制器的安裝，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`。控制器會監視自訂資源的建立、修改和刪除，並透過 App Mesh API 對對應的 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 “路由”、、、、the section called “虛擬路由器” (包括the section called “路由”) 資源進行變更。要了解更多信息或對控制器做出貢獻，請參閱GitHub項目。

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 — 特使使使用 App Mesh 控制平面中定義的配置來確定將應用程序流量發送到何處。
- App Mesh Proxy 路由管理員 — 更新網繭網路命名空間中的`iptables`規則，以透過 Envoy 路由輸入和輸出流量。此容器會在 Pod 內以 Kubernetes 初始化容器執行。

必要條件

- 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。如果您沒有現有叢集，請參閱 Amazon EKS 使用者指南中的 Amazon EKS 入門指南。如果您在 Amazon EC2 上執行自己的 Kubernetes 叢集，請確保 Docker 已向使用者映像所在的 Amazon ECR 儲存庫進行身份驗證。如需詳細資訊，請參閱 Amazon 彈性容器登錄使用者指南中的特使映像、登錄身份驗證，以及 Kubernetes 文件中的從私有登錄提取映像。
- App Mesh 支援使用 DNS 或兩者註冊的 Linux 服務。 AWS Cloud Map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已命名為`serviceA`、`serviceB`，`serviceBv2`且所有服務都可透過名為的命名`apps.local`空間來探索。

  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。
- 已安裝 1.18.116 或更新 AWS CLI 版本，或是 2.0.38 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，則可以使用說明來安裝`kubectl`和設定檔案。`kubeconfig`
- 已安裝 Helm 3.0 版或更新版本。如果您沒有安裝頭盔，請參閱 Amazon EKS 用戶指南中的將頭盔與 Amazon EKS 一起使用。

- Amazon EKS 目前僅支援 IPv4_ONLY IP 偏好設定，因為 Amazon EKS 目前僅支援能夠提供IPv4流量或僅IPv6流量提供服務的網繭。IPv6_ONLY

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

將整合元件安裝到裝載您要與 App Mesh 搭配使用之網繭的每個叢集一次。

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，或者不確定是否已安裝，則可以下載並執行指令碼，以檢查叢集上是否已安裝預先發行版本。

   ```
   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。如需有關升級發行前版本的詳細資訊，請參閱升級於 GitHub。
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。如果尚未安裝 eksctl，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。如果尚未安裝 eksctl，可以按照 Amazon EKS 使用者指南中的安裝或升級 eksctl 中的說明進行安裝。如果您想要使用主控台建立供應商，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 受管政策附加至該角色，然後將其繫結至 `appmesh-controller` Kubernetes 服務帳戶。該角色可讓控制器新增、移除和變更 App Mesh 資源。

> **Note**
>
> 此命令會建立具有自動產生名稱的 AWS IAM 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 建立帳戶，您還需要將角色對應至 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. 部署應用程式網格控制器。如需所有組態選項的清單，請參閱上的組態 GitHub。

   1. 若要為私有叢集部署 App Mesh 控制器，您必須先啟用應用程式網格和服務探索 Amazon VPC 端點至連結的私有子網路。您還需要設置`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。控制器預設會使`public.ecr.aws/xray/aws-xray-daemon:latest`用，因此請將此映像檔提取到本機，然後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 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
      ```

      確認將應用程式部署與虛擬節點或閘道繫結時，是否已成功插入 X-Ray 精靈。

      如需詳細資訊，請參閱 Amazon EKS 使用者指南中的私有叢集。

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

      ```
      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，將下列選項新增至上一個命令--set clusterName=$CLUSTER_NAME來設定叢集名稱。

> **Important**
>
> 如果您的叢集位於me-south-1、、ap-east-1、ap-southeast-3eu-south-1il-central-1、或 Regi af-south-1 ons 中，則您需要將下列選項新增至上一個命令：將*## ID* 和*###############*。
>
> - 對於邊車圖像：
>   - ```
>     --set image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/amazon/appmesh-controller
>     ```
>   - 我-南方 1. 亞馬遜公司/: v1.27.3.0 aws-appmesh-envoy
>   - 阿拉伯東部-亞馬遜網站/: v1.27.3.0-產品 aws-appmesh-envoy
>   - 阿aws-appmesh-envoy拉伯東南部-亞馬遜公司/: v1.27.3.0-產品
>   - 歐aws-appmesh-envoy洲南部亞馬遜公司/: v1.27.3.0-產品
>   - 中央地區-亞馬遜公司/: v1.27.3.0-aws-appmesh-envoy 產品
>   - AFR-南部亚马孙公司网站/: v1.27.3.0 aws-appmesh-envoy
> - 較舊的映像 URI 可以在更改日誌中找到。 GitHub圖像存在的 AWS 帳戶在版本中已更改v1.5.0。圖像的舊版本託管在 Amazon Elastic Kubernetes Service 容器映像註冊表上找到的 AWS 帳戶上。
> - 對於控制器映像檔：
>   - ```
>     --set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
>     ```
>   - 我南部 1. 亞馬遜公司/亞馬遜/應用程序控制器：v1.12.3
>   - 亞馬遜/應用程序控制器：v1.12.3
>   - 亞馬遜/應用程序控制器：v1.12.3

- 歐洲南部 1. 亞馬遜公司/亞馬遜/應用程序控制器：v1.12.3
- 中央-1. 亞馬遜公司/亞馬遜/應用程序控制器：v1.12.3
- AFR-南部 1. 亞馬遜公司/亞馬遜/應用程序控制器：v1.12.3

- 對於邊車初始化映像檔：
  - ```
    --set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
    ```
  - 我南部 1. 亞馬遜公aws-appmesh-proxy-route司/經理：V7-產品
  - 阿拉伯東部 1. 亞馬遜網站/管理員:v aws-appmesh-proxy-route 7-產品
  - 阿拉伯東南部-3. 亞馬遜公aws-appmesh-proxy-route司/-經理：第七集
  - 歐洲南部 1. 亞馬遜公司/經理：第七aws-appmesh-proxy-route集
  - 中央-1. 亞馬遜公aws-appmesh-proxy-route司/-經理：v7-產品
  - AFR-南部-亞馬遜公aws-appmesh-proxy-route司/經理：v7-產品

Important

僅支援 v1.9.0.0 版或更新版本以搭配應用程式網格使用。

10. 確認控制器版本為 `v1.4.0` 或更新版本。您可以檢閱變更登入 GitHub。

```
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

當您在 Kubernetes 中部署應用程式時，也會建立 Kubernetes 自訂資源，以便控制器可以建立對應的 App Mesh 資源。下列程序可協助您部署具有部分功能的 App Mesh 資源。您可以在 App Mesh 逐步解說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。 GitHub

> **Important**
>
> 控制器建立 App Mesh 資源之後，建議您只使用控制器變更或刪除應用程式 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應用程式 Mesh 資源十個小時。您可以將此持續時間設定為較短。如需詳細資訊，請參閱中的組態 GitHub。

部署 App Mesh 資源

1. 建立要將 App Mesh 資源部署到的 Kubernetes 命名空間。

   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. 建立應用程式網格服務網格。

   a. 將下列內容儲存到電腦上名為 `mesh.yaml` 的檔案中。該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. 檢視控制器所建立之 App Mesh 服務網格的詳細資料。

```
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。該文件用於創建命名*my-apps*空間*my-service-a*中命名的 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課程未涵蓋的功能，例如 end-to-end 加密和健康狀態檢查。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料。

> **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 格資源時，將 Kubernetes 命名空間名稱附加至 App Mesh 虛擬節點名稱。新增命名空間名稱是因為在 Kubernetes 中，您可以在不同的命名空間中建立具有相同名稱的虛擬節點，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. 建立應用程式網狀虛擬路由器。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。該文件用於創建一個虛擬路由器，以將流量路由到在上一步中創建my-service-a的名為的虛擬節點。控制器會建立 App Mesh 虛擬路由器並路由資源。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，參照的虛擬節點名稱是 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
```

   (選擇性) 若要查看可在上述規格中設定之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
```

   若要查看可在上述規格中設定之路由的所有可用設定，請執行下列指令。

```
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。您可以指定 `my-service-a-virtual-router_my-appsname`，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，因為路由名稱對於虛擬路由器而言是唯一的。

```
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. 建立應用程式網格虛擬服務。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。雖然名稱對 App Mesh 無關緊要，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。該文件用於創建使用虛擬路由器提供程序的虛擬服務，將流量路由到名為`my-service-a`在上一個步驟中創建的虛擬節點。`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 中建立之虛擬服務資源的詳細資料。Kubernetes 控制器在 App Mesh 格中建立虛擬服務時，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"
    }
}
```

雖然本教程未涵蓋，但控制器也可以部署 App Mesh the section called “虛擬閘道” 和the section called “路由”. 如需透過控制器部署這些資源的逐步解說，請參閱設定輸入閘道，或包含資源的範例資訊清單 GitHub。

步驟 3：建立或更新服務

您想要與 App Mesh 搭配使用的任何網繭都必須新增 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，只串流其自己應用程式網格虛擬節點的組態。

   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、將您在上一步中建立的政策附加至該角色、建立 Kubernetes 服務帳戶，然後將該政策繫結至 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 建立帳戶，您還需要將角色對應至 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。如果尚未安裝`eksctl`，可以按照 Amazon EKS 使用者指南中的安裝或升級`eksctl`中的說明進行安裝。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，就像在的子步驟中所做3的the section called “步驟 2：部署 App Mesh 資源”一樣。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，以便將附屬容器自動新增至網繭，並重新部署網繭。

   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非虛擬節點，則Deployment資訊清單應僅包含 Envoy 容器。如需要使用之影像的更多資訊，請參閱〈〉Envoy。有關 GitHub Manfest 的範例，請參閱（詳見）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。如果您將範例服務部署到 Fargate，則該`envoy`容器已由控制器新增至網繭，但該`proxyinit`容器不是。

4. (選擇性) 安裝附加元件，例如 Prometheus、Grafana AWS X-Ray、積格和資料多。如需詳細資訊，請參閱上的應用程式 Mesh 附加元件 GitHub 和《App Mesh 使用者指南》的「可觀察性」一節。

> **Note**
>
> 如需 App Mesh 的更多範例和逐步解說，請參閱 App M esh 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
```

如果您為示例服務創建了 Fargate 配置文件，請將其刪除。

```
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
```

(選擇性) 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
```

開始使用 AWS App Mesh 和 Amazon EC2

本主題可協助您搭 AWS App Mesh 配在 Amazon EC2 上執行的實際服務搭配使用。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，請假設您有具有下列特性的應用程式：

- 由兩個名為serviceA和的服務組成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% 的流量傳送serviceA到serviceB和 25% 的流量，以serviceBv2驗證serviceBv2是否沒有錯誤，然後再將 100% 的流量傳送serviceA至該流量。
- 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。一旦所有流量被發送到serviceBv2，你想要停止serviceB。
- 您不需要針對實際服務變更任何現有的應用程式程式碼或服務探索註冊，以符合先前的需求。

為了滿足您的需求，您決定使用虛擬服務、虛擬節點、虛擬路由器和路由來建立 App Mesh 服務網格。實施網格後，您將更新服務以使用 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。

必要條件

App Mesh 支援使用 DNS (或兩者) 註冊的 Linux 服務。 AWS Cloud Map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。

如果您尚未執行服務，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 Linux 執行個體使用者指南中的教學課程：開始使用 Amazon EC2 Linux 執行個體的執行個體。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 或更高 AWS CLI 版本或 2.0.38 或更高版本來完成以下步驟。如果使用 AWS CLI，請使用指`aws --version`令來檢查已安裝的 AWS CLI 版本。如果您沒有安裝 1.18.116 或更高版本或 2.0.38 或更高版本，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 開啟 App Mesh 主控台首次執行精靈。
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 AWS Cloud Map 探索實際服務。虛擬節點會在連接埠 80 上使用 HTTP/2 通訊協定監聽流量。也支援其他通訊協定，以及運作狀態檢查。您可以在稍後的步驟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. 對於服務探索方法，請選擇 DNS 並輸入 **serviceb.apps.local** DNS 主機名稱。
3. 在監聽器組態下，選擇 http2 做為通訊協定，然後在連接埠中輸入**80**。
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 檔案作為輸入，使用create-virtual-node指令建立虛擬節點。

```
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。虛擬路由器會在連接埠 80 上使用 HTTP/2 通訊協定監聽流量。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。添加`serviceBv2`虛擬節點後，權重將在稍後的步驟中。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「監聽器組態」下，選擇 http2 做為「通訊協定」，並指**80**定連接埠
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. 對於目標組態下的虛擬節點名稱，**100**為權重選擇`serviceB`並輸入。
6. 在比對模型組態下，選擇一個方法。
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 檔案作為輸入，使用create-virtual-router指令建立虛擬路由器。

```
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指令建立之虛擬服務的設定。

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

檢閱您使用describe-virtual-node指令建立之虛擬節點的設定。

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

檢閱您使用describe-virtual-router指令建立的虛擬路由器的設定。

```
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。對於serviceA虛擬節點，請設定的後端serviceb.apps.local。來自serviceA虛擬節點的所有輸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。
- 建立一個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。將所有流量傳送到之後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. 在監聽器組態中，選取 http2 做為通訊協定，然後在連接埠中輸入**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**serviceA**入虛擬節點名稱。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. 對於在新後端下輸入虛擬服務名稱，輸入**serviceb.apps.local**。
10. 在 [監聽器組態] 下，選擇 http2 做為 [通訊協定]，輸入**80**連接埠，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 [權重] 的值變更`serviceB`為**75**。
14. 選擇添加目標，`serviceBv2`從下拉列表中選擇，然後將重量的值設置為**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. 輸入**servicea.apps.local**虛擬服務名稱，選取提供者的虛擬節點，為虛擬節點選取`serviceA`虛擬節點，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：

- 授權您與每個服務一起部署的 Envoy Proxy，以讀取一或多個虛擬節點的組態。如需如何授權 Proxy 的詳細資訊，請參閱特使代理授權。
- 若要更新您現有的服務，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 政策附加到允許其從 Amazon ECR 讀取的角色，並且僅將特定 App Mesh 虛擬節點的組態連接到該角色。

   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* 替換為您的帳戶 ID 和*#### -2* 替換為您在其中創建網格的區域。

   ```
   {
       "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原則附加至角色，以便該角色只能從 App Mesh 讀取`serviceB`虛擬節點的設定。

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

   d. 將受`AmazonEC2ContainerRegistryReadOnly`管政策附加到角色，以便它可以從 Amazon ECR 提取特使容器映像。

```
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. 根據您的作業系統說明文件，AWS CLI 在執行個體上安裝 Docker 和執行個體。
6. 向您希望 Docker 客戶端從中提取映像的區域中的特使 Amazon ECR 存儲庫進行身份驗證。

   - 除了me-south-1、、、ap-east-1ap-southeast-3、eu-south-1和之外的所有區域af-south-1。il-central-1您可以將 *us-west-2* 取代為任何受支援的區域me-south-1，除了、ap-east-1、ap-southeast-3eu-south-1、il-central-1和。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，在執行個體上啟動 App Mesh Envoy 容器，視您要從哪個區域提取映像檔而定。*apps* 和 *serviceB* 值是案例中定義的網格和虛擬節點名稱。此資訊告訴代理要從 App Mesh 讀取哪些虛擬節點組態。若要完成此案例，您還需要針對託管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-1ap-southeast-3、eu-south-1和之外的所有區域af-south-1。il-central-1您可以將區###取代為任何支援的區域，除了me-south-1、、ap-east-1ap-southeast-3eu-south-1il-central-1、和 af-south-1「地區」以外。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:v1.27.3.0-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:v1.27.3.0-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:v1.27.3.0-prod
```

> **Note**
>
> 該APPMESH_RESOURCE_ARN屬性需要特使圖像的版本1.15.0或更高版本。如需詳細資訊，請參閱 Envoy。

> **Important**
>
> 僅支援 v1.9.0.0 版或更新版本以搭配應用程式網格使用。

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 M esh 範例儲存庫。

App Mesh esh esh 控制

這是AWS App Mesh 的實驗性公開路線圖。該藍圖可讓客戶了解我們即將推出的產品和優先事項，這有助於客戶規劃 future 如何使用 App Mesh。此庫包含了我們正在處理之內容的相關資訊，可以讓所有AWS客戶能夠直接提供意見回饋。

App Mesh 路線圖

App Mesh

您可以在下列儲存庫AWS App Mesh中找到顯示的 end-to-end 逐步解說，以及與各種AWS服務整合的程式碼範例：

App Mesh 示例

App Mesh

App Mesh 由下列元素組成：

- 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

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取器。如果命名空間選擇器是空的，它會選擇所有的命名空間。若要限制命名空間，請使用標籤將 App Mesh 資源與建立的網格相關聯。

AWS Management Console

使用建立服務網格AWS Management Console

1. 在 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. (選擇性) 選擇允許外部流量。根據預設，網狀中的 Proxy 只會在彼此之間轉送流量。如果您允許外部流量，網狀中的 Proxy 也會將 TCP 流量直接轉送至未使用網狀中定義之 Proxy 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]，控制網狀內的流量應使用哪個 IP 版本。根據預設，App Mesh 會使用各種 IP 版本。

> **Note**
>
> 網格會將 IP 偏好設定套用至網狀內的所有虛擬節點和虛擬閘道。您可以在建立或編輯節點時設定 IP 偏好設定，在個別虛擬節點上覆寫此行為。無法覆寫虛擬閘道上的 IP 偏好設定，因為無論網格上設定何種喜好設定，允許它們監聽 IPv4 和 IPv6 流量的虛擬閘道的組態都是相同的。

- 預設
  - 特使的 DNS 解析器更喜歡IPv6並回落IPv4。
  - 我們使用返回的IPv4地址（AWS Cloud Map如果可用），並回到使用該IPv6地址。
  - 為本機應用程式建立的端點會使用IPv4位址。
  - 特使監聽器綁定到所有IPv4地址。
- IPv6 偏好
  - 特使的 DNS 解析器更喜歡IPv6並回落IPv4。
  - 返回的IPv6地址將被AWS Cloud Map使用（如果可用），並回到使用該IPv4地址
  - 為本機應用程式建立的端點會使用IPv6位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。
- IPv4 偏好
  - 特使的 DNS 解析器更喜歡IPv4並回落IPv6。
  - 我們使用返回的IPv4地址（AWS Cloud Map如果可用），並回到使用該IPv6地址。
  - 為本機應用程式建立的端點會使用IPv4位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。

- 僅適用於 IPv6
  - 特使的 DNS 解析器僅使用IPv6。
  - 只會使用傳回AWS Cloud Map的IPv6地址。如果AWS Cloud Map返回IPv4地址，則不會使用 IP 地址，並將空的結果返回給特使。
  - 為本機應用程式建立的端點會使用IPv6位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。
- 僅限 IPv4
  - 特使的 DNS 解析器僅使用IPv4。
  - 只會使用傳回AWS Cloud Map的IPv4地址。如果AWS Cloud Map返回IPv6地址，則不會使用 IP 地址，並將空的結果返回給特使。
  - 為本機應用程式建立的端點會使用IPv4位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。

6. 選擇 Create mesh (建立網格) 以完成。
7. (選擇性) 與其他帳戶共享網格。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，請參閱使用共用的網格。

AWS CLI

若要使用建立網面AWS CLI。

使用下列指令建立服務網格 (使用您自己的指令取代*##*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"
        }
    }
}
```

如需有關使用 App Mesh 建立網格的AWS CLI詳細資訊，請參閱AWS CLI參照中的建立網格指令。

刪除網格

AWS Management Console

使用刪除虛擬閘道AWS Management Console

1. 在 https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要刪除的網格。會列出您擁有且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

若要使用刪除網面AWS CLI

1. 使用下列指令刪除網格 (用您自己的網格取代*##*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"
        }
    }
}
```

如需有關使用 App Mesh 刪除網格的AWS CLI詳細資訊，請參閱AWS CLI參照中的刪除網格指令。

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

若要使用建立虛擬服務AWS Management Console

1. 在 https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要建立虛擬服務的網格。會列出您擁有且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，但建議您使用目標實際服務的服務探索名稱 (例如)`my-service.default.svc.cluster.local`，以便更輕鬆地將虛擬服務與實際服務建立關聯。如此一來，您就不需要變更程式碼以引用不同於當前引用的名稱。您指定的名稱必須解析為非迴路 IP 位址，因為應用程式容器必須能夠成功解析名稱，才能將要求傳送至 Envoy Proxy。您可以使用任何非迴路 IP 位址，因為應用程式或 Proxy 容器都不會與此 IP 位址通訊。Proxy 會透過您在 App Mesh 中為其設定的名稱與其他虛擬服務進行通訊，而不是透過名稱解析的 IP 位址進行通訊。

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連線到虛擬節點，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**
     >
     > 即使您無法透過 App Mesh API 定義此類政策，App Mesh 也可能會為您在 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的特使路由重試原則。如需詳細資訊，請參閱預設路由重試原則。

   - 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

若要使用建立虛擬服務AWS CLI。

使用以下命令和輸入 JSON 文件創建具有虛擬節點提供者的虛擬服務（用您自己的值替換##值）：

1. ```
   aws appmesh create-virtual-service \
   --cli-input-json file://create-virtual-service-virtual-node.json
   ```
2. 示例 create-virtual-service-virtual節點的內容：

   ```
   {
       "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"
    }
}
```

如需有關使用 App Mesh 建立虛擬服務的AWS CLI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引用的虛擬服務，您需要先刪除閘道路由。

AWS Management Console

若要刪除虛擬服務AWS Management Console

1. 在 https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要刪除虛擬服務的網格。會列出您擁有且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. 選擇要刪除的虛擬服務，然後單擊刪除在右上角。您只能刪除您的帳戶被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

若要刪除虛擬服務除AWS CLI

1. 使用以下命令刪除您的虛擬服務（用您自己的值替換*##*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"
    }
}
```

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

虛擬閘道

虛擬閘道允許網狀外部的資源與網狀內部的資源進行通訊。虛擬閘道代表在 Amazon ECS 服務或 Amazon EC2 執行個體中執行的 Envoy 代理。與虛擬節點（代表使用應用程序運行的 Envoy）不同，虛擬網關代表自己部署的 Envoy。

外部資源必須能夠將 DNS 名稱解析為指派給執行 Envoy 的服務或執行個體的 IP 位址。然後，Envoy 可以存取網狀內部資源的所有 App Mesh 組態。在虛擬閘道處理內送要求的組態是使用閘道路由指定的。

> **Important**
>
> 具有 HTTP 或 HTTP2 接聽程式的虛擬閘道會將傳入要求的主機名稱重新寫入閘道路由目標虛擬服務的名稱，而且根據預設，會將來自閘道路由的符合前置詞重新寫入。/例如，如果您已將 Gateway 路由符合前置詞設定為/chapter，且如果傳入要求為/chapter/1，則會將要求重寫為/1。若要設定重寫，請參閱〈從閘道路由建立閘道路由〉一節。
> 當創建一個虛擬網關，proxyConfiguration並且不user應該被配置。

若要完成 end-to-end 逐步解說，請參閱設定輸入閘道。

建立虛擬閘道

> **Note**
>
> 建立虛擬閘道時，您必須新增帶有標籤的命名空間選取器，以識別要將閘道路由與建立之虛擬閘道相關聯的命名空間清單。

AWS Management Console

使用建立虛擬閘道AWS Management Console

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要建立虛擬閘道的網狀。會列出您擁有且已與您共用的所有網格。
3. 選擇左側導覽列中的 [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. (選擇性，但建議使用) 設定用戶端原則預設值。
   a. (選擇性) 如果您希望閘道只能使用傳輸層安全性 (TLS) 與虛擬服務通訊，請選取強制 TLS。
   b. (選擇性) 對於連接埠，指定一或多個要在其上強制與虛擬服務進行 TLS 通訊的連接埠。
   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經存在，而且符合特定需求。如需詳細資訊，請參閱憑證需求。
      - AWS Private Certificate Authority主機 — 選取一或多個現有憑證。
      - 特使秘密發現服務（SDS）託管 — 輸入特使使使用秘密發現服務獲取的秘密名稱。
      - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑。
   d. (選用) 輸入主旨替代名稱。若要新增其他 SAN，請選取新增 SAN。SAN 必須是 FQDN 或 URI 格式。
   e. (選擇性) 選取提供用戶端憑證和下列其中一個選項，以在伺服器要求時提供用戶端憑證並啟用相互 TLS 驗證。若要進一步了解相互 TLS，請參閱應用程式網狀相互 TLS 驗證文件。
      - 特使秘密發現服務（SDS）託管 — 輸入特使使使用秘密發現服務獲取的秘密名稱。

- 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑以及私密金鑰。如需完整內容，請逐 end-to-end 步瞭解如何使用使用本機檔案加密的範例應用程式部署網格，請參閱開啟使用檔案提供的 TLS 憑證設定 TLS GitHub。

7. (選擇性) 若要設定記錄，請選取記錄。輸入您希望使用者使用的 HTTP 存取記錄路徑。我們建議您使用該`/dev/stdout`路徑，以便您可以使用 Docker 日誌驅動程序將特使日誌導出到服務，例如 Amazon CloudWatch 日誌。

> **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. 設定監聽器。

   a. 選取通訊協定，並指定 Envoy 接聽流量的連接埠。http 偵聽器允許連接轉換到網絡套接字。您可以按一下新增監聽程式來新增多個監聽器。刪除按鈕將刪除該監聽器。

   b. (選擇性) 啟用連線集區

      連線共用限制虛擬閘道 Envoy 可以同時建立的連線數。它旨在保護您的 Envoy 執行個體免於連線不堪重負，並可讓您根據應用程式的需求調整流量塑型。

      您可以為虛擬閘道接聽程式設定目的地端連線集區設定。預設情況下，App Mesh 會將用戶端連線集區設定設定設定為無限，以簡化網格組態。

      > **Note**
      >
      `connectionPool`與`connectionPool`連接埠對應通訊協定必須相同。如果您的監聽器協定為`grpc`或`http2`，請`maxRequests`僅指定。如果您的監聽器協定為`http`，您可以同時指定`maxConnections`和`maxPendingRequests`。

      - 對於連線數目上限，請指定輸出連線的數目上限。
      - 針對請求數量上限，指定虛擬閘道 Envoy 可以建立的 parallel 請求數量上限。
      - (選擇性) 對於擱置要求上限，請指定 Envoy 佇列的連線數目上限之後的溢位要求數目。預設值為 2147483647。

   c. (選擇性) 如果您要設定監聽器的健全狀況檢查，請選取啟用健全狀況檢查。

健全 Health 檢查原則是選擇性的，但是如果您為健全 Health 況原則指定任何值，則必須指定狀況臨界值、健全狀況檢查間隔、健全狀況檢查通訊協定、逾時期間和不良狀況臨界值的值。

- 在 Health 檢查通訊協定中，選擇通訊協定 如果您選擇 grpc，則您的服務必須符合 GRPC 運作 Health 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 Health 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，指定收到運作狀態檢查回應所要等候的毫秒數。
- 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. (選擇性) 如果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。

- 在模式中，選取要在監聽器上設定 TLS 的模式。
- 針對憑證方法，選擇下列其中一個選項。憑證必須符合特定需求。如需詳細資訊，請參閱憑證需求。
  - AWS Certificate Manager主機 — 選取現有的憑證。
  - 特使秘密發現服務（SDS）託管 — 輸入特使使使用秘密發現服務獲取的秘密名稱。
  - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結和私密金鑰檔案的路徑。
- (選擇性) 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 驗證。若要進一步了解相互 TLS，請參閱應用程式網狀相互 TLS 驗證文件。
  - 特使秘密發現服務（SDS）託管 — 輸入特使使使用秘密發現服務獲取的秘密名稱。
  - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑。
- (選用) 輸入主旨替代名稱。若要新增其他 SAN，請選取新增 SAN。SAN 必須是 FQDN 或 URI 格式。

9. 選擇 [建立虛擬閘道] 以完成。

AWS CLI

若要使用建立虛擬閘道AWS CLI。

使用以下命令創建一個虛擬網關並輸入 JSON（用您自己的值替換*##*值）：

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

2. 範例 create-virtual-gateway .json 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 App Mesh 建立虛擬閘道的AWS CLI詳細資訊，請參閱AWS CLI參考中的create-virtual-gateway命令。

部署虛擬閘道

部署僅包含特使容器的亞馬遜 ECS 或 Kubernetes 服務。您也可以在 Amazon EC2 執行個體中部署 Envoy 容器。如需詳細資訊，請參閱應用程式網狀和 Amazon EC2 入門。如需有關如何在 Amazon ECS 上部署的詳細資訊，請參閱開始使 App Mesh 和 Amazon ECS 或開始使AWS App Mesh 和 Kubernetes 部署到 Kubernetes。您必須將APPMESH_RESOURCE_ARN環境變數設定為，mesh/*mesh-name*/virtualGateway/*virtual-gateway-name*且不得指定 Proxy 組態，這樣 Proxy 的流量就不會重新導向至自身。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。

我們建議您部署容器的多個執行個體，並設定 Network Load Balancer，以平衡執行個體的流量。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)、建立外部 Load Balancer (Kubernetes) 或教學課程：提高應用程式在 Amazon EC2 上的可用性。您也可以在 App Mesh 範例中找到更多範例和逐步解說。

啟用特使的代理授權。如需詳細資訊，請參閱特使代理授權。

刪除虛擬閘道

AWS Management Console

使用刪除虛擬閘道AWS Management Console

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要從中刪除虛擬閘道的網狀。會列出您擁有且已與您共用的所有網格。
3. 選擇左側導覽列中的 [虛擬閘道]。
4. 選擇您要刪除的虛擬閘道，然後選取 [刪除]。如果虛擬閘道具有任何關聯的閘道路由，則無法刪除該虛擬閘道。您必須先刪除任何關聯的閘道路由。您只能刪除您的帳戶被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. 使用以下命令刪除虛擬閘道（用您自己的值替換*##*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"
    }
}
```

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

路由

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://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要建立 G道路由。會列出您擁有且已與您共用的所有網格。
3. 選擇左側導覽列中的 [虛擬閘道]。
4. 選擇要與新閘道路由建立關聯的虛擬閘道。如果沒有列出，那麼您需要先建立一個。您只能為虛擬閘道建立閘道路由，而您的帳戶會列為資源擁有者。
5. 在「閘道路由」表格中，選擇「建立閘道路由」。
6. 對於閘道路由名稱，請指定要用於閘道路由的名稱。
7. 對於網關路由類型，請選擇 http，http2 或 grpc。

8. 選取現有的虛擬服務名稱。如果沒有列出，那麼您需要先建立一個。
9. 選擇與虛擬服務提供者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）具有多個接聽程式時，需要虛擬服務提供者連接埠。
10. (選擇性) 針對「優先順序」，指定此閘道路由的優先順序。
11. 對於比對組態，請指定：

    - 如果選取的類型為 http/http2：
      - (選擇性) 方法-指定要在內送 http/http2 要求中比對的方法標頭。
      - (選擇性) 連接埠相符-符合傳入流量的連接埠。如果此虛擬閘道具有多個接聽程式，則需要連接埠相符。
      - (選擇性) 精確 /尾碼主機名稱-指定應與傳入要求相符的主機名稱，以路由傳送至目標虛擬服務。
      - (選擇性) 首碼/正則程式/Regex 路徑-比對 URL 路徑的方法。
        - 前綴匹配-通過閘道路由匹配的請求被重寫為目標虛擬服務的名稱/，並默認情況下將匹配的前綴重寫為。視您設定虛擬服務的方式而定，它可以使用虛擬路由器，根據特定的首碼或標頭，將要求路由到不同的虛擬節點。

          > **Important**
          >
          > - 您不能指定/aws-appmesh*或/aws-app-mesh*為前綴匹配。這些前綴保留供 future App Mesh 內部使用。
          > - 如果定義了多個閘道路由，則會將要求與具有最長前置詞的路由相符。例如，如果存在兩個閘道路由，其中一個具有前置詞/chapter，另一個具有前綴為/，則的請求www.example.com/chapter/將與具有/chapter前綴的閘道路由進行匹配。

          > **Note**
          >
          > 如果您啟用基於路徑/前綴的匹配，則 App Mesh 會啟用路徑標準化（規範化路徑和 merge_slashes），以最大程度地減少路徑混淆漏洞的可能性。
          > 當參與要求的各方使用不同的路徑表示時，就會發生路徑混淆弱點。

        - 精確匹配-精確參數禁用路由的部分匹配，並確保僅在路徑與當前 URL 完全匹配時返回路由。

- 正則表達式匹配-用於描述多個 URL 實際識別網站上單個頁面的模式。
- (選擇性) 查詢參數-此欄位可讓您比對查詢參數。
- (選擇性) 標頭-指定 http 和 http2 的標頭。它應該與傳入的請求相匹配，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) 完全/尾碼相符-指定應與傳入要求相符的主機名稱，以路由傳送至目標虛擬服務。
  - grpc 服務名稱-grpc 服務充當您應用程式的 API，並使用定義 ProtoBuf。

> **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 future App Mesh 內部使用。

  - (選擇性) 中繼資料-指定 grpc 的中繼資料。它應該匹配傳入的請求，以路由到目標虛擬服務。

12. （可選）對於重寫配置：

- 如果選取的類型為 http/http2：
  - 如果「字首」為選取的相符類型：
    - 覆寫主機名稱的自動重新寫入-依預設，主機名稱會重新寫入目標虛擬服務的名稱。
    - 覆寫前綴的自動重寫-當打開時，「前綴重寫」會指定重寫前綴的值。
  - 如果「精確路徑」為選取的相符類型：
    - 覆寫主機名稱的自動重寫-依預設，主機名稱會重寫為目標虛擬服務的名稱。
    - 路徑重新寫入-指定重寫路徑的值。沒有預設路徑。
  - 如果正則表達式路徑是選定的匹配類型：
    - 覆寫主機名稱的自動重寫-依預設，主機名稱會重寫為目標虛擬服務的名稱。
    - 路徑重新寫入-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新寫入-依預設，主機名稱會重新寫入目標虛擬服務的名稱。

13. 選擇 [建立閘道路由] 以完成。

AWS CLI

若要使用建立閘道路由AWS CLI。

使用以下命令和輸入 JSON 創建網關路由（用您自己的值替換*##*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. 範例 create-gateway-route .json 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 for App Mesh 建立閘道路由的AWS CLI詳細資訊，請參閱AWS CLI參考中的create-gateway-route命令。

刪除 Gate路由

AWS Management Console

使用刪除 Gate路由AWS Management Console

1. 在 https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。
2. 選擇您要從中刪除 G道路由。會列出您擁有且已與您共用的所有網格。
3. 選擇左側導覽列中的 [虛擬閘道]。
4. 選擇您要從中刪除閘道路由的虛擬閘道。

5. 在「閘道路由」表格中，選擇要刪除的閘道路由，然後選取「刪除」。如果您的帳號被列為資源擁有者，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 Gate路由AWS CLI

1. 使用以下命令刪除您的網關路由（用您自己的值替換*##*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"
    }
}
```

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

虛擬節點

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，必須為任務群組指定服務探索方法。您的虛擬節點預期的任何輸入流量都會指定為接聽程式。虛擬節點傳送輸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關聯的 Amazon 資源名稱 (ARN)。在 Amazon ECS 任務定義或 Kubernetes 網繭規格中，將此值設為任務群組特使代理容器的`APPMESH_RESOURCE_ARN`環境變數。例如，該值可以是`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`1.15.0`或更高版本的 Envoy 映像檔。如需有關 App Mesh 特許變數的詳細資訊，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

若要使用建立虛擬節點 AWS Management Console

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要在其中建立虛擬節點的網格。會列出您擁有且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. 在虛擬節點名稱中，輸入虛擬節點的名稱。
6. 對於服務探索方法，請選擇下列其中一個選項：

   - DNS — 指定虛擬節點所代表之實際服務的 DNS 主機名稱。特使代理部署在亞馬遜 VPC 中。代理伺服器會將名稱解析要求傳送至針對 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 位址。如需 VPC DNS 設定的詳細資訊，請參閱搭配 VPC 人雲端使用 DNS。對於 DNS 回應類型 (選用)，請指定 DNS 解析程式傳回的端點類型。Lo@@ ad Balancer 意味著 DNS 解析器返回一組負載平衡的端點。端點意味著 DNS 解析器正在返回所有端點。依預設，會假設回應類型為 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。或者，您也可以選取 [新增列] 並指定 [索引鍵] 和 [值]，以指定 App Mesh 可以查詢AWS Cloud Map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 AWS Cloud Map 的詳細資訊，請參閱《AWS Cloud Map 開發人員指南》。
   - 無 — 如果您的虛擬節點預期不會有任何輸入流量，請選取此選項。

7. IP 版本偏好設定

   透過切換 [覆寫預設 IP 版本行為]，控制網狀內的流量應使用哪個 IP 版本。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
  - 特使的 DNS 解析器更喜歡IPv6並回落。IPv4
  - 我們使用返回的IPv4地址（AWS Cloud Map如果可用），並回到使用該IPv6地址。
  - 為本機應用程式建立的端點會使用IPv4位址。
  - 特使監聽器綁定到所有IPv4地址。
- 首選
  - 特使的 DNS 解析器更喜歡IPv6並回落。IPv4
  - 返回的IPv6地址將被AWS Cloud Map使用（如果可用），並回到使用該IPv4地址
  - 為本機應用程式建立的端點會使用IPv6位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。
- 首選
  - 特使的 DNS 解析器更喜歡IPv4並回落。IPv6
  - 我們使用返回的IPv4地址（AWS Cloud Map如果可用），並回到使用該IPv6地址。
  - 為本機應用程式建立的端點會使用IPv4位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。
- 僅適用於 IPv6
  - 特使的 DNS 解析器僅使用。IPv6
  - 只會使用傳回AWS Cloud Map的IPv6地址。如果AWS Cloud Map返回IPv4地址，則不會使用 IP 地址，並將空的結果返回給特使。
  - 為本機應用程式建立的端點會使用IPv6位址。
  - 特使監聽器綁定到所有IPv4和IPv6地址。
- 僅限 IPv4
  - 特使的 DNS 解析器僅使用。IPv4
  - 只會使用傳回AWS Cloud Map的IPv4地址。如果AWS Cloud Map返回IPv6地址，則不會使用 IP 地址，並將空的結果返回給特使。
  - 為本機應用程式建立的端點會使用IPv4位址。

- 特使監聽器綁定到所有IPv4和IPv6地址。

8. (選擇性) 用戶端原則預設值 — 設定與後端虛擬服務通訊時的預設需求。

> **Note**
>
> - 如果您想要為現有的虛擬節點啟用傳輸層安全性 (TLS)，建議您建立新的虛擬節點，該節點代表與要啟用 TLS 的現有虛擬節點相同的服務。然後使用虛擬路由器逐漸將流量轉移到新的虛擬節點並進行路由。若要取得有關建立路線和調整變化權值的更多資訊，請參閱〈〉路由。如果您使用 TLS 更新現有的流量服務虛擬節點，則下游用戶端 Envoy 代理很可能會在您已更新之虛擬節點的 Envoy Proxy 接收憑證之前收到 TLS 驗證內容。這可能會導致下游特許代理伺服器上的 TLS 交涉錯誤。
> - 必須為使用後端服務虛擬節點所代表之應用程式部署的 Envoy Proxy 啟用代理伺服器授權。我們建議您在啟用 Proxy 授權時，限制只存取此虛擬節點與之通訊的虛擬節點。

- (選擇性) 如果您要求虛擬節點使用傳輸層安全性 (TLS) 與所有後端通訊，請選取強制 TLS。
- (選擇性) 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。如果您未指定任何通訊埠，則會針對所有連接埠強制執行 TLS。
- 對於驗證方法，請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需求。如需詳細資訊，請參閱憑證需求。
  - AWS Private Certificate Authority主機 — 選取一或多個現有憑證。如需完整內容，請逐 end-to-end 步瞭解如何使用 ACM 憑證加密的範例應用程式部署網格，請參閱開啟 GitHub Cer AWStificate Manager 的 TLS 設定 TLS。
  - 特使秘密發現服務（SDS）託管 — 輸入使用秘密發現服務獲取的秘密特使的名稱。
  - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑。如需完整內容，請逐 end-to-end 步瞭解如何使用使用本機檔案加密的範例應用程式部署網格，請參閱開啟使用檔案提供的 TLS 憑證設定 TLS GitHub。
- (選擇性) 輸入主旨替代名稱。若要新增其他 SAN，請選取新增 SAN。SAN 必須是 FQDN 或 URI 格式。
- (選擇性) 選取提供用戶端憑證和下列其中一個選項，以在伺服器要求時提供用戶端憑證並啟用相互 TLS 驗證。若要進一步了解相互 TLS，請參閱應用程式網狀相互 TLS 驗證文件。
  - 特使秘密發現服務（SDS）託管 — 輸入使用秘密發現服務獲取的秘密特使的名稱。

- 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑以及私密金鑰。

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

   - 為虛擬節點通訊的虛擬服務輸入 App Mesh 虛擬服務名稱或完整的 Amazon 資源名稱 (ARN)。
   - (選擇性) 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。
     - (選擇性) 如果您要求虛擬節點使用 TLS 與所有後端通訊，請選取強制 TLS。
     - (選擇性) 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。如果您未指定任何通訊埠，則會針對所有連接埠強制執行 TLS。
     - 對於驗證方法，請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需求。如需詳細資訊，請參閱憑證需求。
       - AWS Private Certificate Authority主機 — 選取一或多個現有憑證。
       - 特使秘密發現服務（SDS）託管 — 輸入使用秘密發現服務獲取的秘密特使的名稱。
       - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑。
     - (選擇性) 輸入主旨替代名稱。若要新增其他 SAN，請選取新增 SAN。SAN 必須是 FQDN 或 URI 格式。
     - (選擇性) 選取提供用戶端憑證和下列其中一個選項，以在伺服器要求時提供用戶端憑證並啟用相互 TLS 驗證。若要進一步了解相互 TLS，請參閱應用程式網狀相互 TLS 驗證文件。
       - 特使秘密發現服務（SDS）託管 — 輸入使用秘密發現服務獲取的秘密特使的名稱。
       - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑以及私密金鑰。
   - 若要新增其他後端，請選取 [新增後端介面]。

10. (選擇性) 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。我們建議您使用該`/dev/stdout`路徑，以便您可以使用 Docker 日誌驅動程序將特使日誌導出到服務，例如 Amazon CloudWatch 日誌。

    > **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. 監聽器組態

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

a. 如果您的虛擬節點預期輸入流量，請為接聽程式指定連接埠和通訊協定。http 偵聽器允許連接轉換到網絡套接字。您可以按一下新增監聽程式來新增多個監聽器。刪除按鈕將刪除該監聽器。

b. (選擇性) 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目。它旨在保護您的本機應用程式，避免連線不堪重負，並可讓您根據應用程式的需求調整流量塑型。

您可以設定虛擬節點監聽程式的目的地端連線集區設定值。預設情況下，App Mesh 會將用戶端連線集區設定設定設定為無限，以簡化網格組態。

> **Note**
>
> 連線集區和連接埠對應通訊協定必須相同。如果您的監聽器通訊協定是 tcp，請僅指定 [最大連線]。如果您的監聽器協定是 grpc 或 http2，請僅指定最大請求。如果您的監聽器協定是 http，您可以同時指定 MaxConnection 和 maxPendingRequests.

- 對於連線數目上限，請指定輸出連線的數目上限。
- (選擇性) 對於擱置要求上限，請指定 Envoy 將排入佇列的連線數目上限之後的溢位要求數目。預設值為 2147483647。

c. (選擇性) 啟用離群值偵測

在客戶 Envoy 上應用的異常檢測允許客戶對觀察到的已知錯誤故障的連接採取近乎立即的行動。它是斷路器實作的一種形式，可追蹤上游服務中個別主機的健康狀態。

異常值偵測會動態判斷上游叢集中的端點是否正在執行與其他端點不同，並將其從運作良好的負載平衡集中移除。

> **Note**
>
> 若要有效地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 DNS 服務探索方法的回應類型為LOADBALANCER，則 Envoy Proxy 只會選取單一 IP 位址來路由至上游服務。這會使從一組主機中彈出狀況不良的主機的異常偵測行為無

效。如需 Envoy Proxy 與服務探索類型相關行為的詳細資訊，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。
- 對於離群值偵測間隔，請指定頂出掃描分析之間的時間間隔和單位。
- 對於「底座頂出持續時間」，指定頂出主體的基準時間量和單位。
- 對於頂出百分比，請指定負載平衡集區中可退出的主機的最大百分比。

d. (選擇性) 啟用健全狀況檢查 — 設定健全狀況檢查原則的設定。

健全 Health 檢查原則是選擇性的，但是如果您為健全 Health 況原則指定任何值，則必須指定狀況臨界值、健全狀況檢查間隔、健全狀況檢查通訊協定、逾時期間和不良狀況臨界值的值。

- 在 Health 檢查通訊協定中，選擇通訊協定 如果您選取 grpc，則您的服務必須符合 GRPC Health 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 Health 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. (選擇性) 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。

- 在模式中，選取要在監聽器上設定 TLS 的模式。
- 對於憑證方法，請選取下列其中一個選項。憑證必須符合特定需求。如需詳細資訊，請參閱憑證需求。
  - AWS Certificate Manager主機 — 選取現有的憑證。
  - 特使秘密發現服務（SDS）託管 — 輸入使用秘密發現服務獲取的秘密特使的名稱。

- 本機檔案主控 — 指定部署 Envoy Proxy 之檔案系統上憑證鏈結檔案的路徑以及私密金鑰。

- (選擇性) 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 驗證。若要進一步了解相互 TLS，請參閱應用程式網狀相互 TLS 驗證文件。
  - 特使秘密發現服務（SDS）託管 — 輸入使用秘密發現服務獲取的秘密特使的名稱。
  - 本機檔案主控 — 指定部署 Envoy 之檔案系統上憑證鏈結檔案的路徑。
- (選擇性) 輸入主旨替代名稱。若要新增其他 SAN，請選取新增 SAN。SAN 必須是 FQDN 或 URI 格式。

f. (選擇性) 逾時

> **Note**
>
> 如果您指定的逾時時間大於預設值，請務必設定虛擬路由器和逾時大於預設值的路由。但是，如果您將逾時減少為低於預設值的值，則在 Route 上更新逾時是可選的。如需詳細資訊，請參閱路由。

- 要求逾時 — 如果您為監聽器的協定選取 grpc、http 或 http2，您可以指定要求逾時。預設值為 15 秒。的值會0停用逾時。
- 閒置持續時間 — 您可以指定任何監聽器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

若要使用建立虛擬節點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 (將##值取代為您自己的值)：

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

2. 範例 create-virtual-node-dns .json 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 App Mesh 建立虛擬節點的AWS CLI詳細資訊，請參閱AWS CLI參考中的create-virtual-node指令。

刪除虛擬節點

> **Note**
>
> 如果將虛擬節點指定為任何路由中的目標，或指定為任何虛擬服務中的提供者，則無法刪除該虛擬節點。

AWS Management Console

若要刪除虛擬節點 AWS Management Console

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要從中刪除虛擬節點的網格。會列出您擁有且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，選擇要刪除的虛擬節點，然後選取刪除。若要刪除虛擬節點，您的帳號 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

若要刪除虛擬節點 AWS CLI

1. 使用以下命令刪除虛擬節點（用您自己的值替換##值）：

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

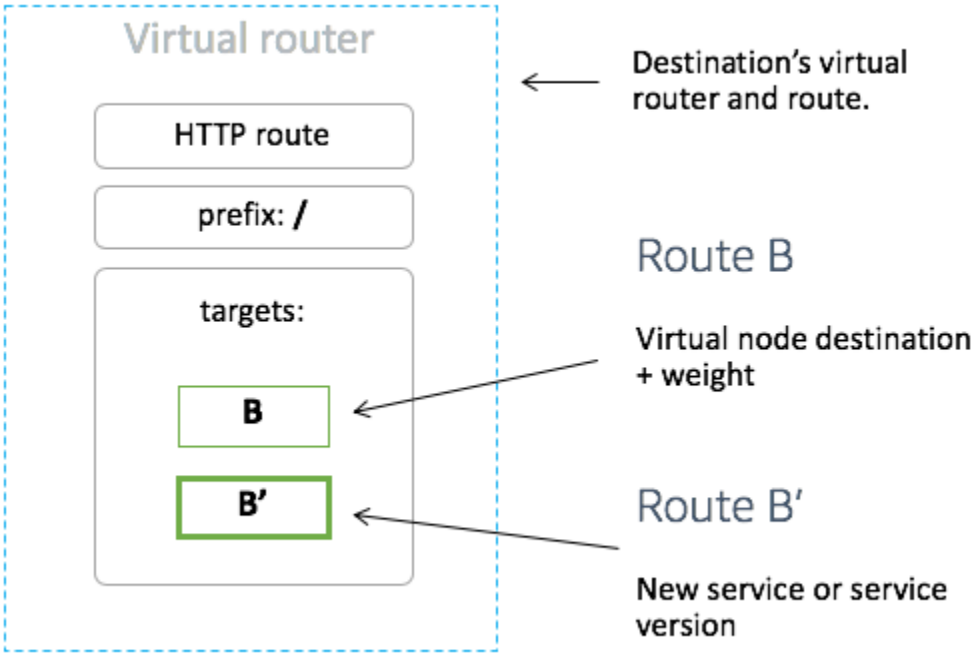
2. 輸出範例：

```
{
    "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"
    }
}
```

如需有關使用 App Mesh 刪除虛擬節點的AWS CLI詳細資訊，請參閱AWS CLI參考中的delete-virtual-node指令。

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。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。



您的虛擬路由器預期的任何入站流量都應指定為接聽程式。

建立虛擬路由器

AWS Management Console

若要建立虛擬路由器AWS Management Console

> **Note**
>
> 建立虛擬路由器時，您必須新增帶有標籤的命名空間選取器，以識別命名空間清單，以將路由與建立的虛擬路由器相關聯。

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要在其中建立虛擬路由器的網格。會列出您擁有且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. (選擇性) 對於「接聽程式」組態，請指定虛擬路由器的連接埠和通訊協定。http監聽器允許連接轉換到網絡套接字。您可以按一下新增監聽程式來新增多個監聽器。刪除按鈕將刪除該監聽器。
7. 選擇 Create virtual router (建立虛擬路由器) 以完成。

AWS CLI

若要使用建立虛擬路由器AWS CLI。

使用以下命令創建一個虛擬路由器並輸入 JSON（用您自己的值替換##值）：

1.
```
aws appmesh create-virtual-router \
     --cli-input-json file://create-virtual-router.json
```
2. 範例 create-virtual-router .json 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 App Mesh 建立虛擬路由器的AWS CLI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**
>
> 如果虛擬路由器具有任何路由或指定為任何虛擬服務的提供者，則無法刪除該路由器。

AWS Management Console

若要刪除虛擬路由器AWS Management Console

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要從中刪除虛擬路由器的網格。會列出您擁有且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] 表格中，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 Mesh 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

若要刪除虛擬路由器AWS CLI

1. 使用以下命令刪除虛擬路由器（用您自己的值替換*##*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"
    }
}
```

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

路由

路由器與虛擬路由器建立關聯。該路由用於匹配虛擬路由器的請求，並將流量分配到其關聯的虛擬節點。如果路由符合請求，它可以將流量分配到一個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。此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. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要在其中建立路由的網格。會列出您擁有且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. 選擇要與新路由相關聯的虛擬路由器。如果沒有列出，那麼您需要先建立一個虛擬路由器。
5. 在 Routes (路由) 表中，選擇 Create route (建立路由)。若要建立路由，您的帳號 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，以及要將流量路由到的虛擬節點。

8. (選擇性) 對於路由優先順序，請指定 0-1000 之間的優先順序以用於您的路由。將依指定值比對路由，0 為最高優先順序。
9. (選擇性) 選擇「其他組態」。從下方的通訊協定中，選擇您為「路由類型」選取的通訊協定，然後視需要在主控台中指定設定。
10. 對於 Target 組態，請選取要將流量路由到的現有 App Mesh 虛擬節點，並指定權重。您可以選擇「新增目標」來新增其他目標。所有目標的百分比之和不得超過 100。如果沒有列出虛擬節點，則您必須先建立一個。如果選取的虛擬節點有多個監聽器，則需要 Target 連接埠。
11. 對於比對組態，請指定：

    比對組態不可用於*tcp*

    - 如果選取的類型為 http/http2：
      - (選擇性) 方法-指定要在內送 http/http2 要求中比對的方法標頭。
      - (選擇性) 連接埠相符-符合傳入流量的連接埠。如果此虛擬路由器具有多個接聽程式，則需要端口匹配。
      - (選擇性) 前綴/完全/正則表達式路徑-匹配 URL 路徑的方法。
        - 前綴匹配-通過閘道路由匹配的請求被重寫為目標虛擬服務的名稱/，並默認情況下將匹配的前綴重寫為。視您設定虛擬服務的方式而定，它可以使用虛擬路由器，根據特定的首碼或標頭，將要求路由到不同的虛擬節點。

          > **Note**
          >
          > 如果您啟用基於路徑/前綴的匹配，則 App Mesh 會啟用路徑標準化（規範化路徑和 merge_sl ash），以最大程度地減少路徑混淆漏洞的可能性。
          > 當參與要求的各方使用不同的路徑表示時，就會發生路徑混淆弱點。

        - 精確匹配-精確參數禁用路由的部分匹配，並確保僅在路徑與當前 URL 完全匹配時返回路由。
        - 正則表達式匹配-用於描述多個 URL 實際識別網站上單個頁面的模式。
      - (選擇性) 查詢參數-此欄位可讓您比對查詢參數。
      - (選擇性) 標頭-指定 http 和 http2 的標頭。它應該與傳入的請求相匹配，以路由到目標虛擬服務。
    - 如果 grpc 是選取的類型：
      - 服務名稱-要符合要求的目的地服務。
      - 方法名稱-要符合要求的目標方法。

- (選擇性) 中繼資料-Match 根據中繼資料的存在指定。所有項目都必須符合才能處理要求。

12. 選取建立路線。

AWS CLI

若要使用建立路線AWS CLI。

使用以下命令和輸入 JSON（用您自己的值替換##值）創建 gRPC 路由：

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

2. 範例 create-route-grpc .json 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"
    }
}
```

如需有關使用 App Mesh 建立路由的AWS CLI詳細資訊，請參閱AWS CLI參照中的 Create-route 指令。

gRPC

(選擇性) 符合

- (選擇性) 輸入目的地服務的服務名稱，以符合要求。若您沒有指定名稱，會符合對任何服務的請求。
- (選擇性) 輸入要符合要求之目的地方法的「方法」名稱。若您沒有指定名稱，對任何方法的請求都符合。如果您指定方法名稱，您必須指定服務名稱。

(選擇性) 中繼資料

選擇 Add metadata (新增中繼資料)。

- (選擇性) 輸入您要路由依據的中繼資料名稱，選取「比對類型」，然後輸入「符合」值。選擇「反轉」將匹配相反的內容。例如，如果您指定的中繼資料名稱`myMetadata`、符合類型為「完全相符」、「相符」值為`123`，並選取「反轉」，則路由就會與任何具有以外任何項目開頭之中繼資料名稱的要求相符`123`。
- (選擇性) 選取「新增中繼資料」以新增最多十個中繼資料項目。

(選擇性) 重試原則

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。重試原則為選用操作，但建議您採用。重試逾時值定義每次重試嘗試的逾時時間 (包括初始嘗試)。如果您未定義重試原則，則 App Mesh 可能會自動為每個路由建立預設原則。如需詳細資訊，請參閱預設路由重試原則。

- 對於「重試逾時」，請輸入逾時持續時間的單位數。如果您選取任何通訊協定重試事件，則需要一個值。
- 對於「重試逾時單位」，請選取一個單位。如果您選取任何通訊協定重試事件，則需要一個值。
- 針對重試次數上限，請輸入請求失敗時的重試次數上限。如果您選取任何通訊協定重試事件，則需要一個值。建議至少使用兩個值。
- 選取一或多個 HTTP 重試事件。我們建議至少選擇流錯誤和閘道錯誤。
- 選取 TCP 重試事件。
- 選取一或多個 gRPC 重試事件。我們建議您至少選取已取消且無法使用。

(選擇性) 逾時

- 預設值為 15 秒。如果您指定了重試原則，則您在此處指定的持續時間應該大於或等於重試持續時間乘以您在重試原則中定義的重試次數上限，以便您的重試原則可以完成。如果您指定的持續時間大於

15 秒，請確定為任何虛擬節點 Target 的監聽器指定的逾時也大於 15 秒。如需詳細資訊，請參閱虛擬節點。

- 的值會0停用逾時。
- 路由可以閒置的時間上限。

HTTP 和 HTTP/2

(選擇性) 符合

- 指定路由應該匹配的前綴。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/路由所有流量。
- (選擇性) 選取方法。
- (選擇性) 選取配置。僅適用於 HTTP2 路由。

（可選）頭

- (選擇性) 選取「新增標頭」。輸入您要依據路由的「表頭」名稱，選取「比對類型」，然後輸入「比對」值。選擇「反轉」將匹配相反的內容。例如，如果您指定一個以前綴為命名`clientRequestId`的標頭123，並選取「反轉」，則路由會與任何具有以外任何項目開頭的標頭的請求進行比對123。
- (選擇性) 選取「新增標頭」。您最多可以新增十個標頭。

(選擇性) 重試原則

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。重試原則為選用操作，但建議您採用。重試逾時值定義每次重試嘗試的逾時時間 (包括初始嘗試)。如果您未定義重試原則，則 App Mesh 可能會自動為每個路由建立預設原則。如需詳細資訊，請參閱預設路由重試原則。

- 對於「重試逾時」，請輸入逾時持續時間的單位數。如果您選取任何通訊協定重試事件，則需要一個值。
- 對於「重試逾時單位」，請選取一個單位。如果您選取任何通訊協定重試事件，則需要一個值。
- 針對重試次數上限，請輸入請求失敗時的重試次數上限。如果您選取任何通訊協定重試事件，則需要一個值。建議至少使用兩個值。
- 選取一或多個 HTTP 重試事件。我們建議至少選擇流錯誤和閘道錯誤。
- 選取 TCP 重試事件。

(選擇性) 逾時

- 要求逾時 — 預設值為 15 秒。如果您指定了重試原則，則您在此處指定的持續時間應該大於或等於重試持續時間乘以您在重試原則中定義的重試次數上限，以便您的重試原則可以完成。
- 閒置持續時間 — 預設值為 300 秒。
- 的值會0停用逾時。

> **Note**
>
> 如果您指定的逾時時間大於預設值，請確定為所有虛擬節點參與者的監聽器指定的逾時值也大於預設值。但是，如果您將逾時減少為低於預設值的值，則可選擇更新虛擬節點上的逾時。如需詳細資訊，請參閱虛擬節點。

TCP

(選擇性) 逾時

- 閒置持續時間 — 預設值為 300 秒。
- 的值會0停用逾時。

刪除路由

AWS Management Console

使用刪除路由AWS Management Console

1. 開啟應用程式網格主控台，位於 https://console.aws.amazon.com/appmesh/。
2. 選擇您要從中刪除路由的網格。會列出您擁有且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器的路由器。
5. 在「路由」表格中，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. 使用以下命令刪除路由（用您自己的值替換##值）：

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

2. 輸出範例：

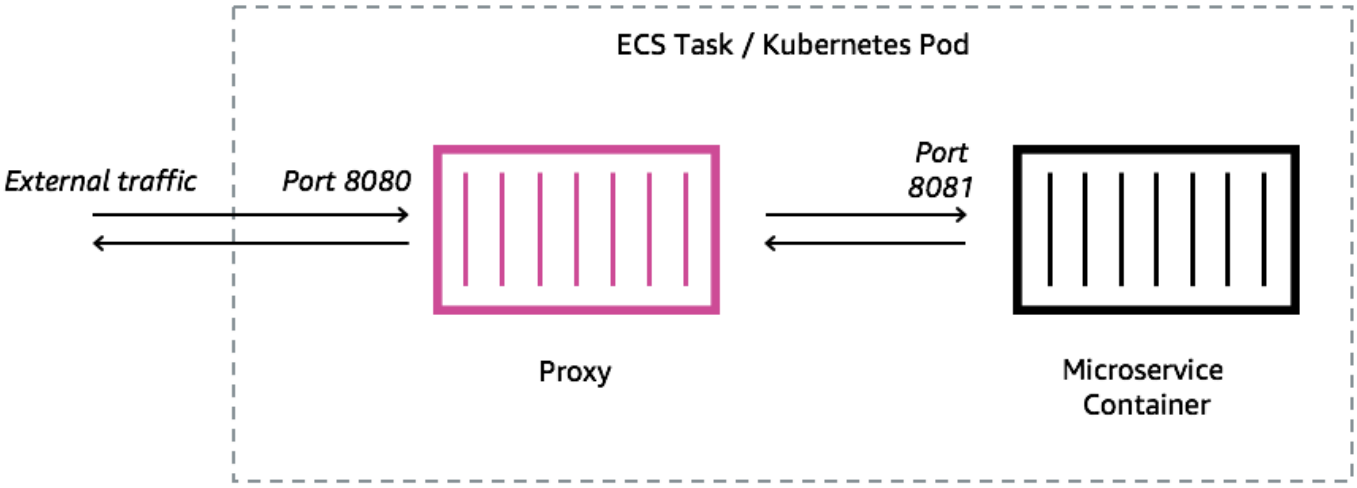
```
{
    "route": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName/route/routeName",
            "createdAt": "2022-04-06T13:46:54.750000-05:00",
            "lastUpdatedAt": "2022-04-07T10:43:57.152000-05:00",
            "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"
    }
}
```

如需有關使用 App Mesh 刪除路由的AWS CLI詳細資訊，請參閱AWS CLI參考資料中的刪除路由指令。

特使形象

AWS App Mesh 是基於特使代理的服務網格。



您必須將特使代理新增至 Amazon ECS 任務、Kubernetes 網繭或 App Mesh 端點所代表的 Amazon EC2 執行個體，例如虛擬節點或虛擬閘道。App Mesh 會出售 Envoy 代理 Docker 容器映像，並驗證此容器映像是否已使用最新的漏洞和性能補丁進行修補。App Mesh 會根據 App Mesh 功能集測試新的特使代理版本，然後再為您提供新的容器映像檔。

您可以從下面的列表中選擇一個區域圖像，也可以從我們名為的公共存儲庫中選擇一個圖像`aws-appmesh-envoy`。

Important

- 從 2023 年 6 月 30 日起，只有特使圖片`v1.17.2.0-prod`或更新版本才能與應用 App Mesh 搭配使用。對於目前使用 v1.17.2.0 版之前的 Envoy 映像檔的客戶，雖然現有使用者仍然相容，但我們強烈建議您移轉至最新版本。
- 最佳作法是，強烈建議您定期將 Envoy 版本升級至最新版本。只有最新的 envoy 版本才能獲得安全補丁，功能發布，性能改進等正式支持。
- 版本`1.17`是特使的重大更新。有關更多詳細信息，請參見更新/遷移到特使 1.17。
- 版本`1.20.0.1`或更高版本`ARM64`兼容。
- 對於`IPv6`支持，需要使用特使版本`1.20`或更高版本。

- 除了me-south-1、、、ap-east-1、ap-southeast-3和以外的所有支援區域af-south-1。eu-south-1 il-central-1您可以將區###取代為me-south-1、、、ap-east-1、ap-southeast-3和以外的任何區域。eu-south-1 il-central-1 af-south-1

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- me-south-1 區域：

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- ap-east-1 區域：

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- ap-southeast-3 區域：

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- eu-south-1 區域：

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- il-central-1 區域：

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- af-south-1 區域：

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

- Public repository

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod
```

> **Note**
>
> 我們建議將 512 個 CPU 單元和至少 64 MiB 的記憶體配置給特使容器。在 Fargate，您可以設定的最低記憶體容量為 1024 MiB 的記憶體。如果容器見解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 容器的資源分配。

> **Note**
>
> 從開始的所有aws-appmesh-envoy映像發行版本v1.22.0.0都構建為無發布性的 Docker 映像。我們進行了這項變更，是為了減少影像大小，並減少影像中未使用的套件中的漏洞暴露。如果您正在構建 aws-appmesh-envoy 圖像之上，並且依賴於某些 AL2 基本軟件包（例如 yum）和功能，那麼我們建議您從aws-appmesh-envoy映像中復制二進製文件以使用 AL2 基礎構建新的 Docker 映像。
> 運行此腳本以生成帶有標籤的自定義碼 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 中對此容器映像的存取由 AWS Identity and Access Management (IAM) 控制。因此，您必須使用 IAM 來驗證您是否擁有 Amazon ECR 的讀取存取權限。例如，使用 Amazon ECS 時，您可以為 Amazon ECS 任務指派適當的任務執行角色。如果您使用 IAM 政策限制對特定 Amazon ECR 資源的存取權限，請確認您允許存取可識別儲存庫的區域特定 Amazon 資源名稱 (ARN)。aws-appmesh-envoy例如，在「us-west-2區域」中，您允許存取下列資源：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。

我們偶爾會發布新的 App Mesh 功能，這些功能取決於尚未合併到上游 Envoy 映像的特使更改。要在上游合併特使更改之前使用這些新的 App Mesh 功能，您必須使用應用程序 Mesh 供應的特使容器映像。如需變更清單，請參閱標Envoy Upstream籤的 App Mesh GitHub 路線圖問題。我們建議您使用 App Mesh Envoy 容器映像作為最佳支援選項。

特使配置變量

使用下列環境變數，為您的 App Mesh 虛擬節點任務群組設定 Envoy 容器。

> **Note**
>
> App Mesh 特使 1.17 不支持特使的 v2 XD API。如果您使用接受特使配置文件的特使配置變量，則必須將它們更新為最新的 v3 X DS API。

必要的變數

所有 App Mesh Envoy 容器都需要下列環境變數。此變數只能與 Envoy 映像檔的版本1.15.0或更新版本搭配使用。如果您使用的是較早版本的影像，則必須改為設定APPMESH_VIRTUAL_NODE_NAME變數。

APPMESH_RESOURCE_ARN

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 或任務群組所代表的虛擬閘道。下列清單包含範例 ARN：

- 虛擬節點 *-ARN#AWN############1111222333###/####/####/virtualNodeName*
- 虛擬閘道器 — *ARN: AW: ######:####:1111222333: ##/####/### #/virtualGatewayName*

使用應用程式網狀預覽通道時，ARN 必須使用 *us-West -2* 區域並使用，而不是使用appmesh-preview。appmesh例如，App Mesh 預覽通道中虛擬節點的 ARN 為arn:aws:**appmesh-preview:us-west-2**:*111122223333*:mesh/*meshName*/virtualNode/*virtualNodeName*。

選擇性變數

下列環境變數對於 App Mesh 特使容器而言是選用的。

`ENVOY_LOG_LEVEL`

指定特使容器的記錄層級。

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

預設：`info`

`ENVOY_INITIAL_FETCH_TIMEOUT`

指定 Envoy 在初始化處理作業期間等待來自管理伺服器的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。設定為時`0`，不會有逾時。

預設：`0`

`ENVOY_CONCURRENCY`

在啟動特使時設置`--concurrency`命令行選項。依預設，不會設定此選項。此選項可從特使版本`v1.24.0.0-prod`或以上版本獲得。

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 的管理介面。

`ENVOY_ADMIN_ACCESS_PORT`

為 Envoy 指定自訂管理連接埠以供監聽。預設：`9901`。

> **Note**
>
> Envoy 管理連接埠應與虛擬 vateway 或虛擬節點上的任何接聽程式連接埠不同

`ENVOY_ADMIN_ACCESS_LOG_FILE`

指定要將 Envoy 存取記錄寫入的自訂路徑。預設：`/tmp/envoy_admin_access.log`。

`ENVOY_ADMIN_ACCESS_ENABLE_IPV6`

將 Envoy 的管理介面切換為接受`IPv6`流量，讓此介面同時接受`IPv4`和`IPv6`流量。根據預設，此旗標設定為 false，且特使僅偵聽流量。`IPv4`此變量只能與特使映像版本 1.22.0 或更高版本一起使用。

代理變數

使用這些環境變數來設定 Envoy 的 AWS App Mesh 代理程式。如需詳細資訊，請參閱特使的應用程式網格代理程式。

APPNET_ENVOY_RESTART_COUNT

指定代理程式在執行中的工作或網繭結束時，重新啟動 Envoy Proxy 處理程序的次數。代理程式也會在每次 Envoy 結束時記錄退出狀態，以簡化疑難排解。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 Proxy 處理序狀態的間隔 (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 Proxy 在程序結束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

啟動代理程式的管理介面伺服器，並將其繫結至 tcp 位址或 unix 通訊端。

有效值：tcp、uds

APPNET_AGENT_HTTP_PORT

指定要在模式中繫結代理程tcp式管理介面的連接埠。確保端口值 > 1024 如果uid！ = 0。確保連接埠小於65535。

預設：9902

APPNET_AGENT_ADMIN_UDS_PATH

指定模式下代理程uds式管理介面的 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」或下列其中一個追蹤驅動程式。

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做為預設精靈端點來啟用 X-Ray 追蹤。若要啟用，請將值設定為1。預設值為 0。

XRAY_DAEMON_PORT

指定要覆寫預設 X-Ray 精靈連接埠的連接埠值：2000。

XRAY_SAMPLING_RATE

指定取樣率，以取代 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。如果已指定，則XRAY_SAMPLING_RULE_MANIFEST會覆寫此值。版本v1.19.1.1-prod及更新版本的 Envoy 映像檔支援此變數。

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 開發人員指南中的抽樣規則。版本v1.19.1.0-prod及更新版本的 Envoy 映像檔支援此變數。

XRAY_SEGMENT_NAME

指定軌跡的段名稱，以取代預設的 X-Ray 段名稱。默認情況下，此值將被設置為mesh/resourceName。使用 Envoy 映像版本v1.23.1.0-prod或更新版本支援此變數。

資料多追蹤變數

下列環境變數可協助您使用 Datadog 代理程式追蹤程式設定應用程式網格。如需詳細資訊，請參閱 Datadog 說明文件中的代理程式組態。

`ENABLE_ENVOY_DATADOG_TRACING`

使用做為預設 Datadog 代理程式端點來啟`127.0.0.1:8126`用資料多格追蹤收集。若要啟用，請將值設定為 1 (預設值為`0`)。

`DATADOG_TRACER_PORT`

指定要覆寫預設 Datadog 代理程式連接埠的連接埠值：8126

`DATADOG_TRACER_ADDRESS`

指定 IP 位址以覆寫預設的 Datadog 代理程式位址：。`127.0.0.1`

`DD_SERVICE`

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

積家追蹤變數

使用下列環境變數，以 Jaeger 追蹤來設定 App Mesh 格。如需詳細資訊，請參閱 Jaeger 文件中的入門指南。版本`1.16.1.0-prod`及更高版本的 Envoy 映像檔支援這些變數。

`ENABLE_ENVOY_JAEGER_TRACING`

使用`127.0.0.1:9411`做為預設 Jaeger 端點來啟用 Jaeger 追蹤收集。若要啟用，請將值設定為 1 (預設值為`0`)。

`JAEGER_TRACER_PORT`

指定要覆寫預設 Jaeger 連接埠的連接埠值：。`9411`

`JAEGER_TRACER_ADDRESS`

指定要覆寫預設 Jaeger 位址的 IP 位址：。`127.0.0.1`

`JAEGER_TRACER_VERSION`

指定收集器是否需要編碼格式`JSON`或`PROTO`編碼格式的追蹤。默認情況下，此值將設置為`PROTO`。使用 Envoy 映像版本`v1.23.1.0-prod`或更新版本支援此變數。

特使追蹤變數

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中，`static_resources`在下設定相關的叢集組態。例如，Zipkin 有一個代管追蹤收集器的叢集名稱`collector_cluster`欄位，而且該叢集需要以靜態方式定義。

DogStatsD 型變數

使用下列環境變數，將應用 App Mesh 設定為 DogStats D。如需詳細資訊，請參閱 DogStatsD 文件。

`ENABLE_ENVOY_DOG_STATSD`

使用`127.0.0.1:8125`做為預設守護程式端點來啟用 DogStats D 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 DogStats D 精靈連接埠。

`STATSD_ADDRESS`

指定 IP 位址值以覆寫預設 DogStats D 精靈 IP 位址。預設：`127.0.0.1`。此變數只能與 Envoy 映像檔的版本`1.15.0`或更新版本搭配使用。

`STATSD_SOCKET_PATH`

指定 DogStats D 守護程序的 unix 域套接字。如果未指定此變數且啟用 DogStats D，則此值預設為的 DogStats D 守護程式 IP 位址連接埠`127.0.0.1:8125`。如果指定的`ENVOY_STATS_SINKS_CFG_FILE`變量包含統計接收器配置，它將覆蓋所有的 DogStats D 變量。使用 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_PREVIEW`

將值設定為以連接1至 App Mesh 預覽通道端點。如需使用應用 App Mesh 預覽通道的詳細資訊，請參閱App Mesh。

`APPMESH_RESOURCE_CLUSTER`

默認情況下，APPMESH_RESOURCE_ARN當 Envoy 在指標和跟踪中引用自身時，App Mesh 會使用您在中指定的資源的名稱。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 Envoy 映像檔的版本1.15.0或更新版本搭配使用。

`APPMESH_METRIC_EXTENSION_VERSION`

將值設定為以啟用「應1用程式網格」度量延伸模組。如需有關使用 App Mesh 量度延伸模組的詳細資訊，請參閱App Mesh 的 App Mesh 的 App Mesh。

`APPMESH_DUALSTACK_ENDPOINT`

將值設定為以連線1至 App Mesh 雙堆疊端點。設定此旗標後，Envoy 會使用我們具備雙堆疊功能的網域。默認情況下，此標誌設置為 false，只連接到我們的IPv4域。此變量只能與特使映像版本 1.22.0 或更高版本一起使用。

特使統計變量

使用下列環境變數，以使用特使統計資料來設定 App Mesh。如需詳細資訊，請參閱特使統計資料文件。

`ENABLE_ENVOY_STATS_TAGS`

啟用應用程式網格定義的標籤appmesh.mesh和appmesh.virtual_node. 如需詳細資訊，請參閱組態。 TagSpecifier在特使文檔中。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，以您自己的檔案路徑覆寫預設 Stats 標籤組態檔案。如需詳細資訊，請參閱組態。 StatsConfig。

> **Note**
>
> 設定包含統計資料篩選器的自訂統計資料組態可能會導致 Envoy 進入不再與全球 App Mesh 狀態正確同步的狀態。這是特使中的一個錯誤。我們的建議是不要在 Envoy 中進行任何統計數據過濾。如果篩選是絕對必要的，我們在藍圖中列出了這個問題的幾個因應措施。

`ENVOY_STATS_SINKS_CFG_FILE`

在 Envoy 容器檔案系統中指定檔案路徑，以您自己的檔案路徑覆寫預設組態。如需詳細資訊，請參閱組態。 StatsSink在特使文檔中。

棄用的變數

環境變量`APPMESH_VIRTUAL_NODE_NAME`和`APPMESH_RESOURCE_NAME`在 Envoy 版本`1.15.0`或更高版本中不再受支持。不過，它們仍然支援現有的網格。不要將這些變數與 Envoy 版本`1.15.0`或更高版本搭配使用，而是用`APPMESH_RESOURCE_ARN`於所有 App Mesh 端點。

特使默認設置由 App Mesh

以下各節提供有關路由重試策略和由 App Mesh 設置的斷路器的 Envoy 默認值的信息。

預設路由重試原則

如果您的帳戶在 2020 年 7 月 29 日之前沒有網格，App Mesh 會在 2020 年 7 月 29 日或之後自動為您帳戶中任何網格中的所有 HTTP、HTTP/2 和 gRPC 要求建立預設的特使路由重試政策。如果您在 2020 年 7 月 29 日之前的帳戶中有任何網格，則不會針對 2020 年 7 月 29 日之前、當日或之後存在的任何特使路線建立預設政策。這是除非你打開一張 AWS 支持票。支援處理票證後，系統會針對 App Mesh 在處理票證當日或之後建立的任何 future Envoy 路由建立預設政策。有關特使路由重試策略的詳細信息，請參閱配置 .route.v3。 RetryPolicy在特使文檔中。

當您建立 App Mesh 路由或為 App Mesh 虛擬服務定義虛擬節點提供者時，App Mesh 會建立特使路由。雖然您可以建立 App Mesh 路由重試原則，但您無法為虛擬節點提供者建立 App Mesh 重試原則。

預設原則無法透過應用程式網格 API 顯示。只有透過 Envoy 才能看到預設政策。若要檢視組態，請啟用管理介面，然後將要求傳送給 Envoy。`config_dump`預設原則包括下列設定：

- 最大重試次數 — 2
- gRPC 重試事件 — `UNAVAILABLE`
- 重試事件 — `503`

> **Note**
>
> 無法建立尋找特定 HTTP 錯誤碼的 App Mesh 路由重試原則。不過，應用程式 Mesh 路由重試原則可以尋找`server-error`或`gateway-error`。這兩個都包括`503`錯誤。如需詳細資訊，請參閱 路由。

- TCP 重試事件 — `connect-failure` 以及 `refused-stream`

> **Note**
>
> 您無法建立尋找其中任一個事件的 App Mesh 路由重試原則。不過，應用程式網格路由重試原則可以尋找`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 (斷開連線/重設/讀取逾時)，Envoy 會嘗試重試。

預設斷路器

當您在 App Mesh 中部署特使時，會針對某些斷路器設定設定設定 Envoy 預設值。如需詳細資訊，請參閱叢集。 CircuitBreakers. 特使文檔中的閾值。透過 App Mesh API 看不到這些設定。這些設置只能通過特使可見。若要檢視組態，請啟用管理介面，然後將要求傳送給 Envoy。`config_dump`

如果您的帳戶在 2020 年 7 月 29 日之前沒有網格，那麼對於您在 2020 年 7 月 29 日或之後建立的網格中部署的每個特使，App Mesh 會透過變更後續設定的特使預設值，有效停用斷路器。如果您在 2020 年 7 月 29 日之前的帳戶中有任何網格，則會為您在 2020 年 7 月 29 日或之後在 App Mesh 中部署的任何特使設定特使預設值，除非您開立支援的票證。 AWS 支援處理票證後，下列 Envoy 設定的 App Mesh 預設值會由您在處理票證之後部署的所有特使上設定：

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

> **Note**
>
> 無論您的特使是否具有特使或 App Mesh 預設斷路器值，您都無法修改這些值。

更新/遷移到特使 1.17

與尖塔的秘密發現服務

如果您將 SPIRE（SPIFE 運行時環境）與應用程序網格一起使用，將信任證書分發到您的服務，請確認您使用的是至少版本`0.12.0`的 SP IRE 代理程序（2020 年 12 月發布）。這是之`1.16`後可以支持 Envoy 版本的第一個版本。

規則運算式變更

從特使開始`1.17`，App Mesh 會將特使設定為預設使用 RE2 規則運算式引擎。這種變化對大多數用戶來說很明顯，但是路由或 Gateway 路由中的匹配不再允許在正則表達式中進行前瞻或反向引用。

正面和負面前瞻

正-正向前瞻是一個括號表達式，開頭為：?=

```
(?=example)
```

這些在進行字符串替換時具有最大的實用性，因為它們允許匹配字符串而不消耗字符作為匹配的一部分。由於 App Mesh 不支援正則表示式字串取代，因此建議您以一般相符項目取代這些字串。

```
(example)
```

負值- 負向前看是以開頭的括號運算式。?!

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

括號表達式用於斷言表達式的該部分與給定的輸入不匹配。在大多數情況下，您可以用零量詞替換它們。

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

如果表達式本身是一個字符類，則可以否定整個類並使用?將其標記為可選。

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

根據您的用例，您可能還可以更改路由以處理此問題。

```
{
    "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"
                        }
                    }
                ]
            }
        }
    }
}
```

第一個路由匹配查找以「前綴」開頭但不是後跟「後綴」的標題。第二條路由的作用是匹配以「前綴」開頭的所有其他標題，包括那些以「後綴」結尾的標題。相反，這些也可以反轉作為消除負面前瞻的一種方式。

```
{
    "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，為以「尾碼」結尾的標頭提供較高的優先順序，並且以「prefix」開頭的所有其他標頭都會在較低優先順序的路由中相符。

反向參考

反向引用是通過重複到前一個括號組來編寫較短表達式的一種方法。他們有這種形式。

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

\後面加上數字的反斜線可做為運算式中第 n 個括號群組的預留位置。在這個例子中，\1被用作第二次寫入(group1)的替代方法。

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

這些可以通過簡單地將反向引用替換為正在引用的組來刪除，如示例中所示。

特使代理程式

該代理是特使映像中的流程管理器，出售應用程 App Mesh。該代理可確保 Envoy 保持運行，保持健康並減少停機時間。它過濾了特使統計信息和輔助數據，以提供特使代理在 App Mesh 中操作的精確視圖。這可以幫助您更快地疑難排解相關錯誤。

您可以使用「代理程式」來設定在 Proxy 運作狀態不良時要重新啟動 Envoy Proxy 的次數。如果發生失敗，當 Envoy 結束時，代理程式會記錄確定性的結束狀態。您可以在疑難排解故障時使用此功能。此代理程式也有助於排除 Envoy 連線，有助於讓您的應用程式對故障更具彈性。

使用下列變數設定 Envoy 的代理程式：

- 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 Proxy 處理程序，直到您在輪詢時將 Proxy 處理程序狀態視為不良時所設定的數目。與在 Proxy 健全狀況檢查失敗的情況下，容器協調器取代工作或網繭相較之下，這有助於減少停機時間。
- PID_POLL_INTERVAL_MS— 配置此變數時，預設值保持為100。當設定為此值時，與透過容器協調器健康狀態檢查的工作或網繭取代相比，您可以在 Envoy 程序結束時更快速地偵測和重新啟動。
- LISTENER_DRAIN_WAIT_TIME_S— 設定此變數時，請考慮為停止工作或網繭設定的容器協調指示器逾時。例如，如果此值大於協調器逾時，則 Envoy Proxy 只能耗盡該持續時間，直到協調器強制停止工作或網繭為止。
- APPNET_AGENT_ADMIN_MODE— 當此變數設定為tcp或時uds，代理程式會提供本機管理介面。此管理介面可做為與 Envoy Proxy 互動的安全端點，並提供下列 API 來進行健康狀態檢查、遙測資料，並概述 Proxy 的作業狀況。
  - GET /status— 查詢特使統計信息並返回服務器信息。

- `POST /drain_listeners`— 排出所有入站聽眾。
- `POST /enableLogging?level=<desired_level>`-更改所有記錄器的特使日誌記錄級別。
- `GET /stats/prometheus`— 顯示 Prometheus 格式的特使統計數據.
- `GET /stats/prometheus?usedonly`-僅顯示特使已更新的統計數據。

如需代理程式組態變數的詳細資訊，請參閱 Envoy 組態變數。

從版本`1.21.0.0`開始，新的AWS App Mesh代理程式包含在 App Mesh 最佳化 Envoy 映像中，不需要在客戶任務或網繭中進行額外的資源配置。

App Mesh 可見性

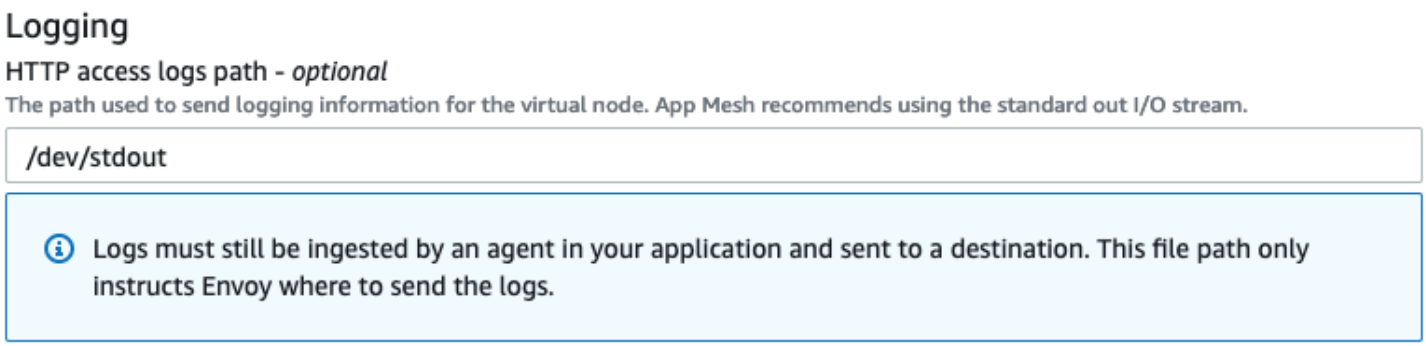
使用 App Mesh 的其中一個好處是對微服務應用程式的可見度更高。App Mesh 能夠使用許多不同的記錄、指標和追蹤解決方案。

Envoy 代理伺服器和 App Mesh 提供以下類型的工具，協助您更清楚地檢視應用程式和代理伺服器：

- 日誌
- Metrics (指標)
- 追蹤

日誌

建立虛擬節點和虛擬閘道時，您可以選擇設定 Envoy 存取記錄。在主控台中，這是在虛擬節點的 [記錄] 區段中，虛擬閘道建立或編輯工作流程。



上述影像顯示 Envoy 存取記錄`/dev/stdout`的記錄路徑。

對於`format`，指定兩種可能的格式之一`text`，`json`或、和樣式。 `json`接受密鑰對並將其轉換為 JSON 結構，然後再將它們傳遞給特使。

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**
>
> 確保檢查您的輸入模式對 Envoy 是否有效，否則 Envoy 將拒絕更新並將最新更改存儲在`error state`.

當您將 Envoy 存取記錄傳送至時`/dev/stdout`，它們會與 Envoy 容器記錄混合在一起。您可以使用標準的 Docker 日誌驅動程序將它們導出到 CloudWatch 日誌存儲和處理服務，例如日誌`awslogs`。如需詳細資訊，請參閱《Amazon ECS ECS ECS 日誌記錄》中的使用 awslog 日誌驅動程式。若只要匯出 Envoy 存取記錄 (並忽略其他 Envoy 容器記錄)，您可以將設定`ENVOY_LOG_LEVEL`為`off`。您可以透過包含格式字串來記錄要求，而不需要查詢字串`%REQ_WITHOUT_QUERY(X?Y):Z%`。如需範例，請參閱`ReqWithoutQuery`格式化程式。如需詳細資訊，請參閱 Envoy 文件中的存取日誌記錄。

在 Kubernetes 上啟用存取日誌

針對 Kubernetes 使用 App Mesh 控制器時，您可以透過將記錄組態新增至虛擬節點規格來設定具有存取記錄的虛擬節點，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"
```

您的叢集必須有記錄轉寄站才能收集這些記錄檔，例如 Fluentd。如需詳細資訊，請參閱將 Fluentd 設定為 DaemonSet 以將記錄檔傳送至 CloudWatch 記錄檔。

Envoy 還將各種調試日誌從其過濾器寫入`stdout`. 這些日誌對於深入了解 Envoy 與 App Mesh 和 service-to-service 流量的溝通非常有用。您可以使用`ENVOY_LOG_LEVEL`環境變數來設定特定的記錄層級。例如，下列文字來自範例偵錯記錄檔，顯示 Envoy 與特定 HTTP 要求相符的叢集。

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

防火鏡和雲觀察

火鏡是一個容器日誌路由器，你可以用來收集日誌亞馬遜 ECS 和AWS Fargate. 您可以在我們的樣AWS品庫中找到使用 Firelens 的示例。

您可以使用 CloudWatch 來收集記錄資訊以及指標。您可以在 App Mesh 文檔 CloudWatch 的導出指標部分中找到有關的更多信息。

使用 Envoy 指標監控您的應用程式

Envoy 將其指標分類為以下主要類別：

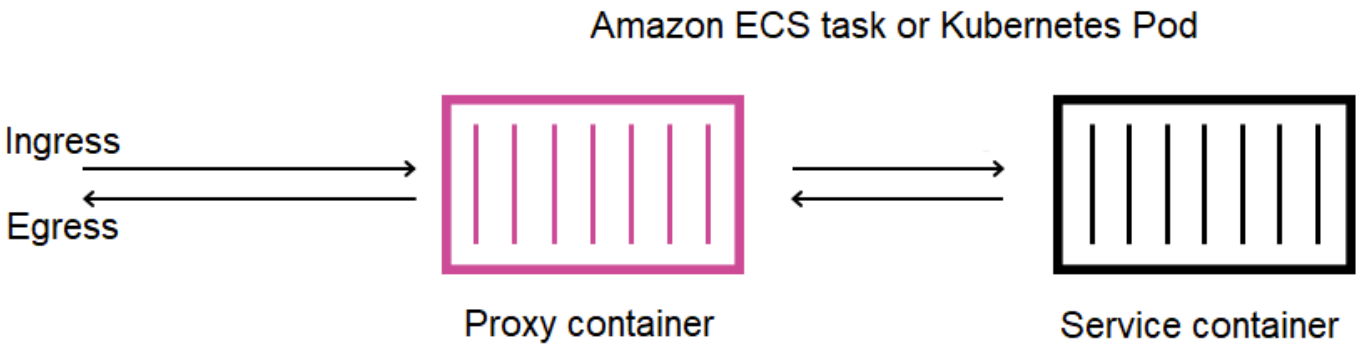
- 下游 — 與進入代理伺服器的連線和要求相關的測量結果。
- 上游 — 與代理伺服器所發出之外送連線和要求相關的測量結果。
- 伺服器 — 描述 Envoy 內部狀態的測量結果。其中包括正常運行時間或分配內存等指標。

在 App Mesh 中，代理攔截上游和下游流量。例如，從您的客戶收到的請求以及服務容器發出的請求都被 Envoy 分類為下游流量。為了區分這些不同類型的上游和下游流量，App Mesh 會根據相對於您的服務的流量方向進一步分類 Envoy 指標：

- Ingress — 與流向服務容器的連線和要求相關的量度和資源。

- 輸出：與從服務容器流動到最終從 Amazon ECS 任務或 Kubernetes 網繭流出的連線和請求相關的度量和資源。

下圖顯示 Proxy 和服務容器之間的通訊。



資源命名慣例

了解 Envoy 如何查看您的網格以及其資源如何映射回您在 App Mesh 中定義的資源非常有用。這些是 App Mesh 配置的主要特使資源：

- 監聽器 — Proxy 監聽下游連線的位址和連接埠。在上一張圖片中，App Mesh 會針對進入 Amazon ECS 任務或 Kubernetes 網繭的流量建立輸入接聽程式，以及用於流量離開服務容器的輸出接聽程式。
- 叢集 — 代理伺服器連線和路由傳送流量的具名上游端點群組。在 App Mesh 中，您的服務容器會表示為叢集，以及您的服務可連線到的所有其他虛擬節點。
- 路線 (Routes)-這些對應於您在網格中定義的路線。它們包含代理匹配請求以及發送請求的目標集群的條件。
- 端點和叢集載入指派 — 上游叢集的 IP 位址。當您將虛擬節點用AWS Cloud Map作服務探索機制時，App Mesh 會將探索到的服務執行個體做為端點資源傳送至 Proxy。
- 機密 — 包括但不限於您的加密金鑰和 TLS 憑證。當AWS Certificate Manager作用戶端和伺服器憑證的來源使用時，App Mesh 會將公用和私有憑證作為秘密資源傳送至您的 Proxy。

App Mesh 使用一致的配置來命名 Envoy 資源，您可以使用這些資源與您的網格建立關聯。

了解偵聽器和集群的命名方案對於了解 Envoy 在 App Mesh 中的指標非常重要。

監聽器名稱

監聽程式是使用下列格式如下：

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

您通常會看到在 Envoy 中設定的下列監聽器：

- `lds_ingress_0.0.0.0_15000`
- `lds_egress_0.0.0.0_15001`

使用 Kubernetes CNI 外掛程式或 IP 表格規則，Amazon ECS 任務或 Kubernetes 網繭中的流量會導向至連接埠15000和15001。App Mesh 會使用這兩個偵聽程式設定 Envoy，以接受入口 (傳入) 和出口 (傳出) 流量。如果您的虛擬節點上沒有設定監聽器，就不會看到輸入接聽程式。

叢集名稱

大部分叢集使用下列格式如下：

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

您的服務與每個服務通訊的虛擬節點都有自己的叢集。如前所述，App Mesh 會為 Envoy 旁邊執行的服務建立叢集，以便代理伺服器可以將輸入流量傳送至該伺服器。

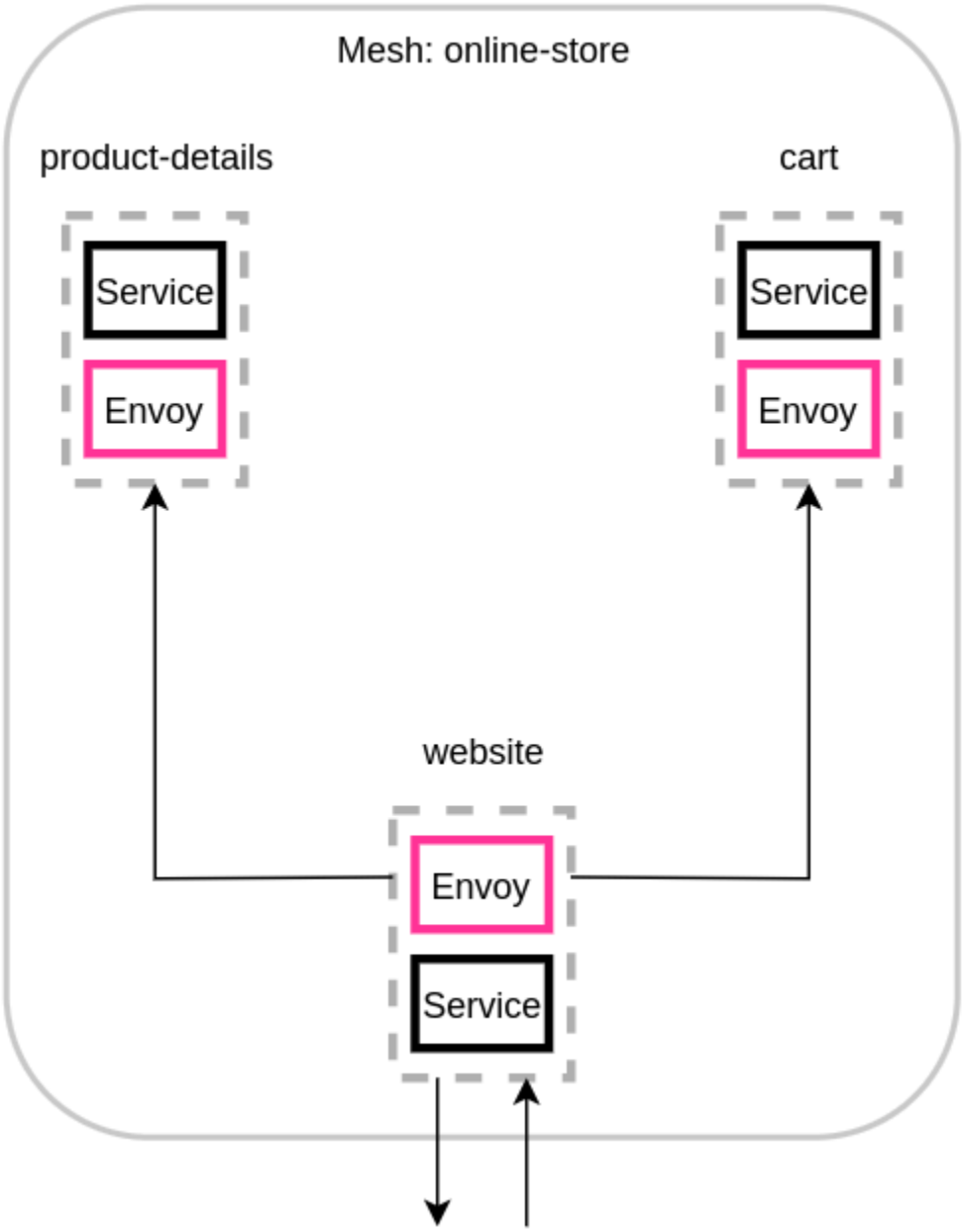
例如，如果您有一個名為監聽連接埠上`my-virtual-node`的 http 流量的虛擬節點，`8080`而該虛擬節點位於名為的網格中`my-mesh`，則 App Mesh 會建立名為的叢集`cds_ingress_my-mesh_my-virtual-node_http_8080`。此叢集可做為進入服務容器之流量`my-virtual-node`的目的地。

App Mesh 也可能會建立下列類型的額外特殊叢集。這些其他叢集不一定與您在網格中明確定義的資源相對應。

- 用來連線其他服AWS務的叢集。預設情況下，此類型可讓您的網格連線到大多數AWS服務：`cds_egress_<mesh name>_amazonaws`。
- 用來執行虛擬閘道路由的叢集。這通常可以安全地忽略：。
  - 對於單個監聽器：`cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<protocol>_<port>`
  - 對於多個偵聽器：`cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>`
- 當您使用 Envoy 的秘密探索服務擷取密碼時，您可以定義端點的叢集 (例如 TLS)`static_cluster_sds_unix_socket`。

範例 App Mesh

為了說明 Envoy 中可用的指標，以下示例應用程序具有三個虛擬節點。網狀中的虛擬服務、虛擬路由器和路由可以忽略，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。



我們建議將環境變數新增ENABLE_ENVOY_STATS_TAGS=1至網狀中執行的 Envoy 代理容器。這會將下列指標維度新增至 Proxy 發出的所有指標：

- `appmesh.mesh`

- appmesh.virtual_node
- appmesh.virtual_gateway

這些標籤設定為網狀、虛擬節點或虛擬閘道的名稱，以允許使用網狀中的資源名稱篩選指標。

資源名稱

網站虛擬節點的代理具有以下資源：

- 兩個用於入口和出口流量的接聽程式：
  - 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
- 網站服務容器的輸入叢集：
  - cds_ingress_online-store_website_http_8080

監聽器度量範例

- listener.0.0.0.0_15000.downstream_cx_active與 Envoy 的作用中輸入網路連線數目。
- listener.0.0.0.0_15001.downstream_cx_active與 Envoy 的作用中輸出網路連線數目。此計數包含應用程式與外部服務所建立的連線。
- listener.0.0.0.0_15000.downstream_cx_total-與使者的輸入網路連線總數。
- listener.0.0.0.0_15001.downstream_cx_total-與使者的出口網路連線總數。

如需完整的監聽器測量結果集，請參閱 Envoy 說明文件中的統計資料。

叢集指標範例

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

下列叢集測量結果使用的格式cluster.<cluster name>.<metric name>。這些指標名稱在應用程式範例中是唯一的，由網站 Envoy 容器發出：

- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_total`-網站和產品詳細信息之間的連接總數。
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_connect_fail`— 網站與產品詳細資料之間失敗的連線總數。
- `cluster.cds_egress_online-store_product-details_http_8080.health_check.failure`— 網站和產品詳細資料之間失敗的運作狀態檢查總數。
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_total`-網站和產品詳細信息之間提出的請求總數。
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_time`-網站和產品詳細信息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`-網站從產品詳細信息收到的 HTTP 2xx 響應的數量。

如需完整的 HTTP 指標集，請參閱 Envoy 文件中的統計資料。

管理伺服器度量

特使還發出與其與 App Mesh 控制平面的連接相關的指標，該控制平面充當 Envoy 的管理服務器。我們建議您監視其中一些測量結果，以便在 Proxy 長時間從控制平面不同步處理時通知您。失去與控制平面的連線或失敗的更新會導致 Proxy 從 App Mesh 接收新的設定，包括透過 App Mesh API 進行的網狀變更。

- `control_plane.connected_state`當代理連接到 App Mesh 時，此度量設置為 1，否則為 0。
- `*.update_rejected`Envoy 拒絕的組態更新總數。這些通常是由於用戶配置錯誤所致。例如，如果您將 App Mesh 設定為從 Envoy 無法讀取的檔案讀取 TLS 憑證，則會拒絕包含該憑證路徑的更新。
  - 對於更新的偵聽器被拒絕，統計數據將是`listener_manager.lds.update_rejected`。
  - 對於集群更新被拒絕，統計數據將是`cluster_manager.cds.update_rejected`。
- `*.update_success`App Mesh 對您的代理進行的成功配置更新次數。其中包括啟動新 Envoy 容器時傳送的初始組態承載。
  - 對於監聽器更新成功，統計數據將是`listener_manager.lds.update_success`。
  - 對於群集更新成功，統計數據將是`cluster_manager.cds.update_success`。

如需管理伺服器測量結果集，請參閱 Envoy 文件中的管理伺服器。

匯出指標

Envoy 在自己的操作以及入站和出站流量的各種維度上發出許多統計信息。要了解有關特使統計信息的更多信息，請參閱 Envoy 文檔中的統計信息。這些測量結果可透過 Proxy 的管理連接埠 (通常為) 上的`/stats`端點取得`9901`。

根據您使用的是單個還是多個偵聽器，`stat`前綴將會有所不同。以下是一些示例來說明差異。

> **Warning**
>
> 如果您將單個偵聽器更新為多個偵聽器功能，則由於下表所示的更新後的 stat 前綴，您可能會面臨突破性更改。
> 我們建議您使用特使圖像`1.22.2.1-prod`或更高版本。這可讓您在 Prometheus 端點中看到類似的度量名稱。

單一接聽程式 (SL)/具有「輸入」偵聽程式前綴的現有統計資料	多個監聽器（ML）/帶有「入口的新統計信息。 <protocol>。<port>"監聽器前置詞	

`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`	

`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`	

單一接聽程式 (SL)/具有「輸入」偵聽程式前綴的現有統計資料	多個監聽器（ML）/帶有「入口的新統計信息。 <protocol>。 <port>"監聽器前置詞	
	`.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`	

如需有關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。如需關於管理介面的詳細資訊，請參閱啟用特使代理管理介面。

Prometheus App Mesh 與 Amazon EKS

Prometheus 是一種開放原始碼監控和警示工具組。它的功能之一是指定發出可供其他系統使用的指標的格式。如需有關 Prometheus 的詳細資訊，請參閱 Prometheus 文件中的概觀。Envoy 可以通過傳遞參數通過其統計終點發出其指標`/stats?format=prometheus`。

對於使用特使映像版本 v1.22.2.1-prod 的客戶，還有兩個額外的維度可指示輸入接聽程式的特定統計資料：

- `appmesh.listener_protocol`
- `appmesh.listener_port`

以下是 Prometheus 現有統計數據與新統計數據的比較。

- 具有「輸入」監聽器前綴的現有統計信息

```
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「入口」的新統計數據。 <protocol>。 <port>「+ 應用網格特使圖片 v1.22.2.1-產品或更高版本

```
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「入口」的新統計數據。 <protocol>。 <port>「+ 自定義特使圖像構建

```
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
```

對於多個偵聽器，cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>特殊群集將是監聽器特定的。

- 具有「輸入」監聽器前綴的現有統計信息

```
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「入口」的新統計數據。 <protocol>。 <port>」

```
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 存儲庫添加到頭盔：

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

2. 安裝 App Mesh theus App Prometheus

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

Prometheus 範例

以下是建立 Prometheus 永久儲存的範例。`PersistentVolumeClaim`

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system \
--set retention=12h \
--set persistentVolumeClaim.claimName=prometheus
```

使用 Prometheus 的演練

- 具有 EK 的 App Mesh — 可觀察性：Prometheus

通過 Amazon EKS 了解有關 Prometheus 和 Prometheus 的更多信息

- Prometheus 文檔
- EKS- 使用 Prometheus 的控制平面度量

CloudWatch App Mesh

CloudWatch 從亞馬遜 EKS 發送特使統計信息

您可以將 CloudWatch 代理程式安裝到叢集，並將其設定為從 Proxy 收集指標子集。如果您還沒有 Amazon EKS 叢集，則可以使用逐步解說：開啟 Amazon EKS 的 App Mesh 中的步驟建立叢集 GitHub。您可以遵循相同的逐步解說，在叢集上安裝範例應用程式。

若要為叢集設定適當的 IAM 許可並安裝代理程式，請依照使用 Prometheus 指標集合安裝 CloudWatch 代理程式中的步驟執行。默認安裝包含一個 Prometheus 抓取配置，該配置可提取有用的特使統計數據子集。如需詳細資訊，請參閱 App Mesh 的 Prometheus 指標的 App Mesh 的指標。

若要建立設定為顯示代理程式正在收集的指標的 App Mesh 自訂 CloudWatch 儀表板，請依照檢視您的 Prometheus 指標教學課程中的步驟執行。當流量進入 App Mesh 應用程式時，您的圖形將開始填入對應的量度。

篩選指標 CloudWatch

App Mesh 度量延伸模組提供一系列有用的量度，可讓您深入瞭解您在網格中定義的資源行為。由於CloudWatch代理支持抓取 Prometheus 指標，因此您可以提供抓取配置，以選擇要從特使中提取並發送到的指標 CloudWatch。

您可以在我們的指標擴展演練中找到使用 Prometheus 抓取指標的示例。

CloudWatch 範例

您可以在我AWS們的範例儲存庫 CloudWatch 中找到的範例組態。

使用逐步解說 CloudWatch

- 在我們的 App Mesh 研討會中新增監控和記錄功能。
- 具有 EK 的 App Mesh-可觀察性： CloudWatch
- 在 ECS 上使用應用程式網格的指標延伸

App Mesh 的 App Mesh 的 App Mesh

Envoy 生成數百個分解為幾個不同維度的指標。指標與應用程式網格相關的方式並不簡單。對於虛擬服務，沒有機制可以確定哪個虛擬服務正在與給定的虛擬節點或虛擬閘道進行通信。

App Mesh 度量擴充功能可增強在網狀中執行的 Envoy 代理。此增強功能可讓 Proxy 發出知道您所定義資源的其他度量。這一小部分的其他量度將有助於您更深入地了解您在 App Mesh 中定義的那些資源的行為。

若要啟用 App Mesh 度量延伸模組，請將環境變數設定`APPMESH_METRIC_EXTENSION_VERSION`為`1`。

```
APPMESH_METRIC_EXTENSION_VERSION=1
```

如需 Envoy 組態變數的詳細資訊，請參閱特使配置變量。

與入站流量相關的指標

- **`ActiveConnectionCount`**

  - envoy.appmesh.ActiveConnectionCount— 作用中 TCP 連線的數目。
  - 尺寸 — 網格, VirtualNode, VirtualGateway
- **NewConnectionCount**
  - envoy.appmesh.NewConnectionCount— TCP 連線的總數。
  - 尺寸 — 網格, VirtualNode, VirtualGateway
- **ProcessedBytes**
  - envoy.appmesh.ProcessedBytes— 從下游用戶端傳送和接收的 TCP 位元組總數。
  - 尺寸 — 網格, VirtualNode, VirtualGateway
- **RequestCount**
  - envoy.appmesh.RequestCount— 已處理的 HTTP 要求數目。
  - 尺寸 — 網格, VirtualNode, VirtualGateway
- **GrpcRequestCount**
  - envoy.appmesh.GrpcRequestCount— 已處理的 GPRC 請求數目。
  - 尺寸 — 網格, VirtualNode, VirtualGateway

與輸出流量相關的量度

根據輸出量度來自虛擬節點還是虛擬閘道，您會在輸出量度上看到不同的維度。

- **TargetProcessedBytes**
  - envoy.appmesh.TargetProcessedBytes— 傳送至特使上游目標和接收的 TCP 位元組總數。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode
- **HTTPCode_Target_2XX_Count**
  - envoy.appmesh.HTTPCode_Target_2XX_Count— 發送給特使上游目標的 HTTP 要求數目，而產生 2xx HTTP 回應。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode
- **HTTPCode_Target_3XX_Count**

  - envoy.appmesh.HTTPCode_Target_3XX_Count— 發送給特使上游目標的 HTTP 要求數目，而產生了 3xx HTTP 回應。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode
- **HTTPCode_Target_4XX_Count**
  - envoy.appmesh.HTTPCode_Target_4XX_Count— 發送給特使上游目標的 HTTP 要求數目，而造成 4xx HTTP 回應。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode
- **HTTPCode_Target_5XX_Count**
  - envoy.appmesh.HTTPCode_Target_5XX_Count— 發送給特使上游目標的 HTTP 要求數目，而產生 5xx HTTP 回應。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode
- **RequestCountPerTarget**
  - envoy.appmesh.RequestCountPerTarget— 傳送至 Envoy 上游目標的要求數目。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode
- **TargetResponseTime**
  - envoy.appmesh.TargetResponseTime— 從對 Envoy 上游的目標發出請求到收到完整回應的時間。
  - 尺寸:
    - 虛擬節點尺寸 — 網格 VirtualNode,, TargetVirtualService, TargetVirtualNode
    - 虛擬閘道維度 — 網格 VirtualGateway、、 TargetVirtualService、 TargetVirtualNode

App Mesh

Datadog 是用於端對端監控、指標和雲端應用程式記錄的監控和安全應用程式。Datadog 讓您的基礎架構、應用程式和協力廠商應用程式完全可以觀察到。

安裝資料多格

- EKS-要使用 EKS 設置數據多，請按照以下數據多文檔中的步驟進行操作。
- ECS EC2-若要使用 ECS EC2 設定資料多，請依照資料多格文件中的下列步驟執行。

進一步了解 Datadog 的詳細資訊

- 資料多文件

追蹤

> **Important**
>
> 若要完全實作追蹤，您需要更新應用程式。
> 要查看所選服務中的所有可用數據，您必須使用適用的庫來檢測應用程序。

使用AWS X-Ray 監 App Mesh

AWSX-Ray 是一項服務，提供可讓您能夠藉此檢視及篩選您的應用程式處理的資料以獲取深入見解，讓您能夠藉此檢視及篩選您的 這些見解可協助您找出最佳化應用程式的問題和機會。您可以查看有關請求和回應的詳細資訊，以及應用程式對其他AWS服務進行的下游呼叫。

X-Ray 與 App Mesh 整合，以管理您的特使微服務。來自 Envoy 的跟踪數據被發送到容器中運行的 X-Ray 守護程序。

使用特定於您語言的 SDK 指南，在您的應用程式程式碼中實作 X-Ray。

透過 App Mesh 進行 X-Ray 追蹤

- 根據服務的類型：
  - ECS- 在 Envoy 代理容器定義中，將`ENABLE_ENVOY_XRAY_TRACING`環境變數設定為`1`，將`XRAY_DAEMON_PORT`環境變數設定為`2000`。

- EKS- 在 App Mesh 控制器配置中，包括`--set tracing.enabled=true`和`--set tracing.provider=x-ray`。
- 在 X-Ray 容器中，暴露端口`2000`並以用戶身份運行`1337`。

X-Ray 範例

亞馬遜 ECS 的特使容器定義

```
      {
        "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
      }
```

更新亞馬遜 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
```

X-Ray 使用 X-Ray 的演練

- AWSX-Ray 監測
- 亞馬遜 EKS App Mesh-可觀測性：X-Ray
- AWS App Mesh工作坊中使用 X-Ray 進行分散式追蹤

了解有關AWS X-Ray 的更多信息

- AWSX-Ray 文件

使用應用程式網格排除AWS X-Ray

- 無法看到我的應用程式的AWS X-Ray 追蹤。

Jaer 的 App Mesh 和 Amazon EKS

Jaeger 是一個開源的，端到端的分佈式跟踪系統。它可用於分析網絡和監視。Jaeger 還可協助您排除複雜的雲端原生應用程式。

要將 Jaeger 實現到您的應用程序代碼中，您可以在 Jaeger 文檔跟踪庫中找到特定於您的語言的指南。

使用頭盔安裝積格

1. 將 EKS 存儲庫添加到頭盔：

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

2. 安裝 App Mesh Jaer

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

積格的例子

以下是建立 Jaeger 永久性儲存裝置的範例。`PersistentVolumeClaim`

```
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
```

使用積格的演練

- 具有 EK 的應用程序網格 — 可觀察性：積格

進一步了解 Jaer 的詳細資訊

- 積格文件

用於追蹤的資料多

Datadog 可用於跟踪以及指標。如需詳細資訊和安裝指示，請在 Datadog 文件中尋找您應用程式語言的特定指南。

App Mesh

App Mesh 讓客戶能夠使用下列工具間接與 API 互動：

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- 適用於 Kubernenetes 的 App Mesh 控制器
- 地形

App MeshAWS CloudFormation

AWS CloudFormation是一項服務，可讓您使用應用程式所需的所有資源建立範本AWS CloudFormation，然後為您配置和佈建資源。它還將配置所有的依賴關係，所以你可以更專注於你的應用程序和更少的管理資源。

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。

App MeshAWS CDK

AWS CDK是一個開發框架，用於使用代碼定義您的雲基礎架構並AWS CloudFormation用於佈建它。AWS CDK支持多種編程語言 TypeScript JavaScript，包括，蟒蛇，Java 和 C＃/.net。

如需AWS CDK搭配 App Mesh 搭配使用的詳細資訊，請參閱AWS CDK文件。

適用於 Kubernenetes 的 App Mesh 控制器

Kubernetes 適用的 App Mesh 控制器可協助您管理 Kubernetes 叢集的 App Mesh 資源，並將側車插入網繭中。此控制器專為搭配 Amazon EKS 使用，可讓您以 Kubernetes 原生的方式管理資源。

如需有關 App Mesh 控制器的詳細資訊，請參閱應用程式網狀控制器文件。

若要查看使用 Kubernetes 應用程式網狀控制器透過 Amazon EKS 實作應用程式網格的指南，請參閱 Amazon EKS 研討會。

App Mesh

Terraform 是一種開放原始碼碼工具。Terraform 可以使用 CLI 管理雲服務，並使用聲明配置文件與 API 進行交互。

若要進一步瞭解如何搭配地形使用應用程式網格，請查看 Terraform 文件。

使用共用的網格

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。

AWS Identity and Access Management帳號可以是網狀資源擁有者、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狀中建立資源。網格擁有者可以與下列類型的網格取用者共用網格：

- AWS 組織內外的特定 AWS Organizations 帳戶
- 之組織內的組織單位AWS Organizations
- 中的整個組織AWS Organizations

有關共用網格的瀏覽，請參閱中的跨帳戶網格穿越 GitHub。 end-to-end

共用網格權限

共用網格具有下列權限：

- 消費者可以列出並描述與帳戶共用的網格中的所有資源。
- 擁有者可以列出並描述帳戶擁有的任何網格中的所有資源。
- 擁有者和取用者可以修改帳號建立的網格中的資源，但無法修改其他其他帳號所建立的資源。
- 取用者可以刪除帳號建立的網格中的任何資源。
- 擁有者可以刪除網格中建立的任何帳號的任何資源。
- 擁有者的資源只能參考同一帳號中的其他資源。例如，虛擬節點只能參照AWS Cloud Map或與虛擬節點擁有者位於相同帳戶中的AWS Certificate Manager憑證。
- 擁有者和消費者可以將 Envoy 代理連接到 App Mesh 作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。
- 擁有者和取用者可以列出標籤，並且可以在帳戶建立的網格中標記/取消標記資源。他們無法在不是由帳戶建立的網格中列出標籤和標記/取消標記資源。

共用網格從帳戶層級允許授權的支援移轉至原則型授權。在此推出之前共享的任何網格都使用先前的策略，並且在此推出後共享的任何網格都使用後一種策略。

透過以原則為基礎的授權，網格會與一組固定的權限共用。系統會選取這些許可以新增至資源政策，也可以根據 IAM 使用者/角色選取選用的 IAM 政策。這些原則中允許的權限交集 (除了任何明確拒絕的明確權限) 決定主參與者對網格的存取權限。

新增資源策略的權限集是固定的，並由AWS Resource Access Manager (AWS RAM) 決定：

- 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

> **Note**
>
> 這個集合會排除某些權限，例如`appmesh:TagResources`。如果需要`appmesh:TagResources`訪問權限，我們可以將您的帳戶映射到帳戶級別允許所有授權策略，直到我們啟動支持為止`appmesh:TagResources`。

共享網格的先決條件

若要共享網格，您必須符合下列先決條件。

- 您必須擁有AWS帳戶中的網格。您無法將已共享給您的網格再共享出去。
- 若要與組織或內的組織單位共享網格，您必須透過啟用共享網格AWS Organizations，您必須透過啟用共享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，該帳戶之間具有共用連線能力，其中包含您要彼此通訊的網狀資源。共用網路連線的一種方法是將您要在網狀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）集成。 AWS RAM是一項服務，可讓您與任何AWS帳戶或透過共用AWS資源AWS Organizations。您可以透過 AWS RAM 建立資源共享，以分享您擁有的資源。資源共享指定要分享的資源，以及共用它們的消費者。消費者可以是個別的 AWS 帳戶，或 AWS Organizations 中的組織單位或整個組織。

如需 AWS RAM 的詳細資訊，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。只能共享自己擁有的網格。若要共享網格，您必須將它新增至資源共享中。資源共享是可讓您在 AWS 帳戶之間分享資源的一種 AWS RAM 資源。資源共享指定要共享的資源，以及共享它們的消費者。若您使用 Amazon Linux 主控台共享網格，可以將該網格新增至現有的資源共享中。若要將網格新增至新的資源共享中，請使用AWS RAM主控台建立資源共享。

如果您是中組織的一分子，AWS Organizations並已啟用與您所屬組織共享的功能，則組織中的消費者便能自動存取所共用的網格。否則，消費者會收到加入資源共享的邀請，並且在接受邀請後便能存取共享的網格。

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主控台共享您擁有的網格

如需指示，請參閱《使用指南》中的〈建立資源共AWS RAM用〉。當您選取資源類型時，請選取網格，然後選取您要共用的網格。如果沒有列出網面，請先建立網面。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。對於此`--resource-arns`選項，指定您要共享的網面的 ARN。

取消共享的網格

當您取消共用網狀時，App Mesh 會停用網格的前消費者進一步存取網格。不過，App Mesh 不會刪除取用者所建立的資源。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 可防止擁有網狀資源的帳戶在取消共用網格後接收組態資訊。App Mesh 也可防止任何其他具有網狀資源的帳戶從未共用的網格接收組態資訊。只有網面的擁有者可以取消共用網面。

若要取消共享您擁有的已共享網格，您必須從資源共享中移除它。您可以使用 AWS RAM 主控台或 AWS CLI 執行這項作業。

使用AWS RAM主控台來取消共享您擁有的已共享網格

如需指示，請參閱AWS RAM使用指南中的更新資源共用。

使用取消共享您擁有的已共享網面AWS CLI

使用 disassociate-resource-share 命令。

點選共用的網格

擁有者和消費者可以使用 Amazon Linux 主控台和來識別共享的網格和網狀資源AWS CLI

使用亞馬遜的 Linux 主控台來識別共享的網格

1. 在 https://console.aws.amazon.com/appmesh/ 開啟 App Mesh 主控台。

2. 從左側導覽中，選取「網面」。每個網格的網格擁有者帳戶 ID 會列在「網格擁有者」欄中。
3. 在左側導覽中，選取 [虛擬服務]、[虛擬路由器] 或 [虛擬節點]。您會看到每個資源的 Mesh 擁有者和資源擁有者的帳號 ID。

使用識別共用網面的步驟AWS CLI

使用`aws appmesh list` *`resource`*指令，例如`aws appmesh list-meshes`。此指令會傳回您擁有的網面和已共享給您的網面。`meshOwner`屬性會顯示的AWS帳號 ID，`meshOwner`而且`resourceOwner`屬性會顯示資源擁有者的AWS帳號 ID。對任何網格資源執行的任何命令都會傳回這些屬性。

您附加至共享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`命令被拒絕訪問。

計費和計量

共享網格無須收費。

實例配額

網格的所有配額也會套用至共用網格，無論是誰在網格中建立資源。只有網格擁有者可以要求增加配額。如需詳細資訊，請參閱App Mesh。該AWS Resource Access Manager服務也有配額。如需詳細資訊，請參閱 Service Quotas。

AWS服務與應用程式網格整合

App Mesh 可使用其他AWS服務，為您的業務挑戰提供額外的解決方案。本主題識別使用應用程式網格來新增功能的服務，或應用 App Mesh 用來執行工作的服務。

內容

建立 App Mesh 資源AWS CloudFormation

App Mesh 已整合AWS CloudFormation，這項服務可協助您建立AWS資源的模型和設定，以減少建立和管理資源和基礎設施的時間。您建立一個描述所有所需之AWS資源的範本 (如 App Mesh 網格)，而AWS CloudFormation負責為您佈建與設定這些資源。

當您使用時AWS CloudFormation，您可以重複使用您的您的資源，重複且一致地設定您的 App Mesh 資源。您只需要描述您的資源一次，然後在多個 AWS 帳戶與區域內重複佈建相同資源即可。

App Mesh 和AWS CloudFormation範本

若要佈建和設定應用程式網格與相關服務的資源，您必須了解的AWS CloudFormation資源。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 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 Designer 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 Designer？。

App Mesh 支援在中建立網格、路由、虛擬節點、虛擬路由器和虛擬服務AWS CloudFormation。如需詳細資訊，包括應用 App Mesh 資源的 JSON 和 YAML 範本範本範例，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》https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

- 《AWS CloudFormation 命令列介面使用者指南》https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/what-is-cloudformation-cli.html

AWSOutposts 上的 App Mesh

AWSOutposts post 會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，您可以使用您在AWS雲端中所使用的相同AWS API、工具及基礎設施。AWSOutpost 上的 App Mesh 適合需要靠近內部部署資料和應用程式執行的低延遲工作負載。有關AWS Outposts 的更多信息，請參閱 AWSOutposts 用戶指南。

先決條件

以下是在AWS Outposts 上使用應用程式網格的先決條件：

- 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 的AWS區域必須支援 Outpost 的區域必須支援AWS App Mesh。如需支援的區域清AWS App Mesh單，請參閱 AWS 一般參考.

限制

以下是在AWS Outposts 上使用應用程序網格的限制：

- AWS Identity and Access Management、、、Classic Application Load Balancer Network Load Balancer、Classic Load Balancer、Classic Load Balancer 和 Amazon Route 53 在AWS區域執行，而不是在 Outposts post。這將提高這些服務和容器之間的延遲。

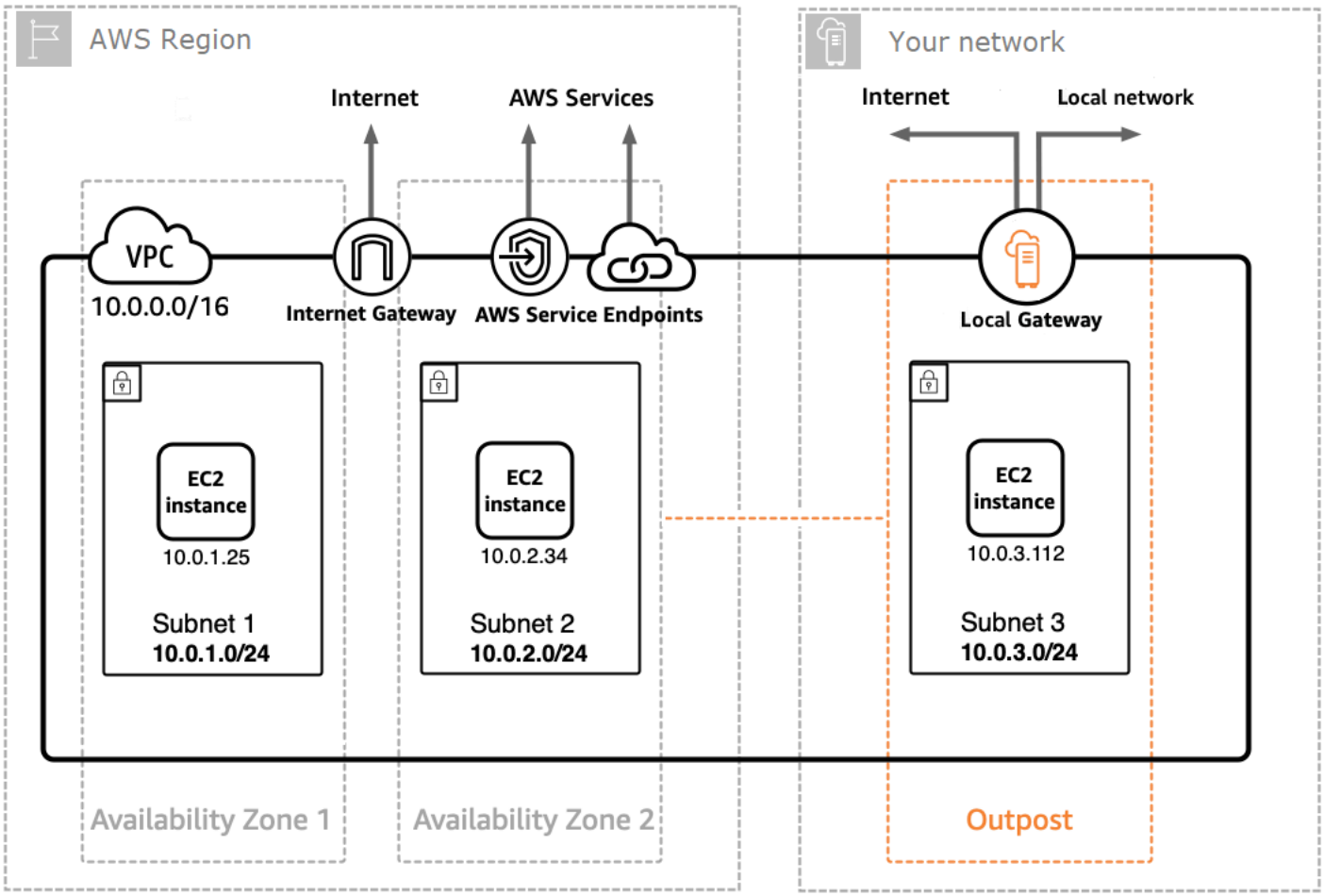
網路連線能力考量

以下是 Amazon EKSAWS Outposts 的網路連線注意事項：

- 如果 Outpost 與其AWS區域之間的網路連線中斷，App Mesh Enpost 的必須繼續運作。不過，在恢復連線之前，您將無法修改服務網格。
- 建議您在 Outpost 與其AWS區域之間提供可靠、高可用性和低延遲的連線能力。

在前哨上創建 App Mesh 特使代理

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。



若要在前哨上建立 App Mesh 特使代理，請將 App Mesh 特使容器映像新增至 Amazon ECS 任務或在前哨站上執行的 Amazon EKS 網繭。有關更多信息，請參閱亞馬遜彈性容器服務開發人員指南中的AWS Outposts 上的亞馬遜彈性容器服務和亞馬遜 EKS 用戶指南中的AWS Outposts 上的亞馬遜彈性Kubernetes 服務。

App Mesh 最佳實務

為了在計劃部署期間以及在某些主機意外遺失期間實現零失敗要求的目標，本主題中的最佳作法會實作下列策略：

- 使用安全的預設重試策略，從應用程式的角度增加要求成功的可能性。如需詳細資訊，請參閱檢測重試的所有路線。
- 將重試要求傳送至實際目的地的可能性最大化，以增加重試要求成功的可能性。如需詳細資訊，請參閱 調整部署速度、在縮放之前向外縮放 及 實作容器運作狀態查。

為了大幅減少或消除失敗，我們建議您在下列所有作法中實作建議。

檢測重試的所有路線

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原則。這將通過重新選擇主機並發送新請求來緩解失敗的請求。對於重試原則，我們建議的值至少為兩個`maxRetries`，並針對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：

- 傳輸協定 —`connection-error`
- HTTP 和 HTTP/2 —`stream-error` 以及`gateway-error`
- gRPC —`cancelled` 以及`unavailable`

其他重試事件需要在 case-by-case 基礎上進行考慮，因為它們可能不安全，例如，如果請求不是冪等的。您將需要考慮並測試`maxRetries`和的值，以`perRetryTimeout`便在請求的最大延遲 (`maxRetries`*`perRetryTimeout`) 與更多重試的成功率提高之間進行適當的折衷。此外，當 Envoy 嘗試連接到不再存在的端點時，您應該預期該請求會消耗完整的端點`perRetryTimeout`。若要設定重試原則，請參閱建立路由並選取您要路由的通訊協定。

> **Note**
>
> 如果您在 2020 年 7 月 29 日當天或之後實作路由，但未指定重試原則，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，自動建立類似先前原則的預設重試原則。如需詳細資訊，請參閱預設路由重試原則。

調整部署速度

使用滾動式部署時，可降低整體部署速度。根據預設，Amazon ECS 會將部署策略設定為至少 100% 的健康狀態任務和 200 百分比的任務總數。在部署時，這會導致兩個高漂移點：

- 在準備好完成請求之前，Envoys 可以看到 100% 的新工作叢集大小 (請參閱實作容器運作狀態查以瞭解緩解措施)。
- 任務終止時，使者可以看到 100% 的舊任務的叢集大小。

使用這些部署條件約束進行設定時，容器協調器可能會進入同時隱藏所有舊目的地並顯示所有新目的地的狀態。由於您的 Envoy 數據通道最終是一致的，因此這可能會導致數據通道中可見的目標集從協調器的角度分歧的時期。為了減輕這種情況，我們建議維持至少 100% 的健康狀態工作，但將工作總數降低到 125%。這將減少分歧並提高重試的可靠性。我們建議您使用下列設定使用下列容器執行下列設定

Amazon ECS

如果您的服務有兩個或三個所需的計數，設置`maximumPercent`為 150%。否則，`maximumPercent`請設定為百分之 125。

Kubernetes

設定您的部署`update strategymaxUnavailable`，設定`maxSurge`為 0% 和 25%。如需有關部署的詳細資訊，請參閱 Kubernetes 部署文件。

在縮放之前向外縮放

向外擴充和擴展都可能導致重試失敗請求的某些可能性。雖然有可減輕向外擴充的工作建議，但擴充的唯一建議是隨時將工作縮放的百分比降到最低。我們建議您在舊任務或部署中進行擴展之前，先使用部署策略來擴展新的 Amazon ECS 任務或 Kubernetes 部署。這種擴展策略可以使您在任務或部署中縮放的百分比降低，同時保持相同的速度。此做法同時適用於 Amazon ECS 任務和 Kubernetes 部署。

實作容器運作狀態查

在向上擴展案例中，Amazon ECS 任務中的容器可能會出現故障，而且初始可能無法回應。我們建議您針對不同的容器執行階段提供下列建議：

Amazon ECS

為了減輕此問題，我們建議您使用容器運作狀態檢查和容器相依性順序，以確保 Envoy 在任何需要輸出網路連線的容器開始之前，都能正常執行且狀態良好 若要在工作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 Kubernetes 的應用程式網格控制器中的AWS Cloud Map執行個體註冊和取消註冊時，不會考慮 Kubernetes 活性和整備程度探查。如需詳細資訊，請參閱 GitHub 問題 #132。

中的安全性 AWS App Mesh

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。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 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。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 負責將設定安全地傳送至本機代理伺服器，包括 TLS 憑證私密金鑰等機密。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您還需要對其他因素負責，包括：
  - 您的資料的敏感性、貴公司的要求，以及適用的法律和法規。
  - App Mesh 資料平面的安全性設定，包括允許流量在 VPC 內的服務之間通過的安全群組的組態。
  - 與 App Mesh 相關聯的運算資源設定。
  - 與您的運算資源相關聯的 IAM 政策，以及允許這些政策從 App Mesh 控制平面擷取哪些設定。

本文件可協助您了解如何在使用 App Mesh 時套用共同的責任模型。下列主題說明如何設定 App Mesh 以符合安全性和合規性目標。您也會學到如何使用其他可 AWS 協助您監視和保護 App Mesh 資源的服務。

App Mesh 安全宗旨

> 客戶應該能夠根據需要調整安全性。平台不應阻止它們更安全。平台功能預設是安全的。

主題

- Transport Layer Security (TLS)
- 交互 TLS 驗證
- 如何與 IAM AWS App Mesh 搭配使用
- 使用記錄 AWS App Mesh API 呼叫 AWS CloudTrail
- AWS App Mesh 中的資料保護
- AWS App Mesh 的合規驗證
- 基礎架構安全性 AWS App Mesh
- AWS App Mesh 中的恢復能力

- 中的配置和漏洞分析 AWS App Mesh

Transport Layer Security (TLS)

在 App Mesh 中，傳輸層安全性 (TLS) 會加密部署在應用程式網格中由網狀端點 (例如和) 在 App Mesh 中表示之計算資源上的 Envoy Proxy 之間的通訊。虛擬節點 虛擬閘道代理伺服器會交涉並終止 TLS。使用應用程式部署 Proxy 時，您的應用程式程式碼不負責交涉 TLS 工作階段。代理伺服器代表您的應用程式交涉 TLS。

App Mesh 可讓您透過下列方式將 TLS 憑證提供給代理伺服器：

- 來自 AWS Certificate Manager (ACM) 的私人憑證，由 AWS Private Certificate Authority (AWS Private CA) 發行。
- 儲存在虛擬節點本機檔案系統上的憑證，由您自己的憑證授權單位 (CA) 發行
- 由密碼探索服務 (SDS) 端點透過本機 Unix 網域通訊端提供的憑證。

特使代理授權必須為由網格端點表示的已部署 Envoy 代理啟用。我們建議您在啟用 Proxy 授權時，將存取限制為只啟用加密的網狀端點。

憑證需求

憑證上的其中一個主體別名 (SAN) 必須符合特定條件，具體取決於探索網狀端點所代表的實際服務的方式而定。

- DNS — 其中一個憑證 SAN 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，或建立具有萬用字元的憑證*.*apps.local*。
- AWS Cloud Map— 其中一個憑證 SAN 必須符合使用該格式*service-name.namespace-name*的 AWS Cloud Map 服務探索設定中提供的值。對於具有 serviceName *mesh-endpoint* 和命名空間名稱 AWS Cloud Map 服務探索設定的應用程式*apps.local*，您可以建立與名稱相符的憑證*mesh-endpoint.apps.local*，或建立具有萬用字元的憑證 *.*apps.local*.

對於這兩種探索機制，如果沒有任何憑證 SAN 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，如用戶端 Envoy 所示。

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

TLS 認證憑證

使用 TLS 驗證時，應用 App Mesh 支援多個憑證來源。

AWS Private CA

憑證必須與將使用憑證的網狀端點儲存在相同的區域和 AWS 帳戶中的 ACM 中。CA 的憑證不需要位於相同的 AWS 帳戶中，但仍需要與網狀端點位於相同的區域中。如果您沒有 AWS 私有 CA，則必須先建立一個，然後才能從中要求憑證。如需有關使用 ACM 向現有 AWS Private CA 使用 ACM 要求憑證的詳細資訊，請參閱要求私有憑證。憑證不能是公用憑證。

您用於 TLS 用戶端原則的私人 CA 必須是根使用者 CA。

若要使用憑證和 CA 來設定虛擬節點 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 權限：

- 對於您新增至接聽程式 TLS 組態的任何憑證，主體必須具有`acm:DescribeCertificate`權限。
- 對於 TLS 用戶端原則上設定的任何 CA，主體必須具有`acm-pca:DescribeCertificateAuthority`權限。

> **Important**
>
> 與其他帳戶共用 CA 可能會將這些帳戶提供非預期的權限給 CA。我們建議您使用以資源為基礎的政策`acm-pca:DescribeCertificateAuthority`，`acm-pca:GetCertificateAuthorityCertificate`限制只存取不需要從 CA 發行憑證的帳戶。

您可以將這些許可新增至附加至主體的現有 IAM 政策，或建立新的主體和政策，然後將該政策附加到主體。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**
>
> 您每月支付每個操作的費用， AWS Private CA 直到刪除它為止。您也需要支付每個月發行的私有憑證和匯出的私有憑證費用。如需詳細資訊，請參閱 AWS Certificate Manager 定價。

當您為網狀端點所代表的 Envoy Proxy 啟用代理授權時，必須為您使用的 IAM 角色指派下列 IAM 許可：

- 對於在虛擬節點的接聽程式上設定的任何憑證，角色必須具有`acm:ExportCertificate`權限。
- 對於 TLS 用戶端原則上設定的任何 CA，角色必須具有`acm-pca:GetCertificateAuthorityCertificate`權限。

檔案系統

您可以使用檔案系統將憑證散發給 Envoy。您可以透過在檔案路徑上使用憑證鏈結和對應的私密金鑰來執行此操作。這樣，可以從 Envoy 側車代理訪問這些資源。

特使的秘密發現服務（SDS）

Envoy 透過秘密探索通訊協定從特定端點擷取 TLS 憑證等機密。如需有關此通訊協定的詳細資訊，請參閱 Envoy 的 SDS 文件。

當 SDS 作為證書和證書鏈的源時，應用程序網格配置 Envoy 代理以使用代理本地的 Unix 域套接字作為秘密發現服務（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**
>
> 使用 Unix 域套接字的本地密碼發現服務在 App Mesh 特使代理版本 1.15.1.0 及更高版本上支持。
> App Mesh 格支援使用 GrPC 的 V2 SDS 通訊協定。

與 SPIRE 運行時環境（SPIRE）集成

您可以使用 SDS API 的任何側車實現，包括現有的工具鏈，如 SPIRE 運行時環境（SPIRE）。SPIRE 旨在在分散式系統中的多個工作負載之間部署相互 TLS 驗證。它會在執行階段驗證工作負載的身分。SPIRE 還提供工作負載特定、短期使用時間，並自動將金鑰和憑證直接旋轉至工作負載。

您應該將 SPIRE 代理配置為特使的 SDS 提供商。允許其直接向 Envoy 提供提供相互 TLS 驗證所需的金鑰材料。在特使代理旁邊的側車中運行 SPIRE 代理。代理程式會根據需要重新產生短暫的金鑰和憑證。當 Envoy 連線至 SPIRE 代理程式所公開的 SDS 伺服器時，代理程式會驗證特使，並決定應提供給特使的哪些服務身分識別和 CA 憑證。

在此程序期間，會輪替服務身分識別和 CA 憑證，並將更新串流回 Envoy。Envoy 立即將它們應用於新的連接，而不會造成任何中斷或停機，並且沒有私鑰接觸文件系統。

App Mesh 如何設定使者以協商 TLS

當決定如何在網狀中設定 Envoys 之間的通訊時，App Mesh 會使用用戶端和伺服器的網狀端點組態。

使用用戶端原則

當用戶端原則強制使用 TLS，且用戶端原則中的其中一個連接埠符合伺服器策略的通訊埠時，用戶端原則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原則符合虛擬節點的伺服器策略，則會使用虛擬閘道用戶端策略中定義的設定，在 Proxy 之間嘗試 TLS 交涉。如果用戶端原則與伺服器策略的連接埠不符，則代理伺服器之間的 TLS 可能會進行交涉，也可能不會交涉，這取決於伺服器策略的 TLS 設定。

沒有用戶端政策

如果用戶端尚未設定用戶端原則，或者用戶端原則與伺服器的連接埠不符，則 App Mesh 將使用伺服器判斷是否從用戶端交涉 TLS，以及如何交涉 TLS。例如，如果虛擬閘道尚未指定用戶端原則，且虛擬節點尚未設定 TLS 終止，則不會在 Proxy 之間交涉 TLS。如果用戶端尚未指定相符的用戶端原則，且伺服器已設定 TLS 模式`PERMISSIVE`，`STRICT`或是 Proxy 會設定為交涉 TLS。視提供 TLS 終止的憑證方式而定，會套用下列其他行為。

- ACM 管理的 TLS 憑證 — 當伺服器使用 ACM 管理的憑證設定 TLS 終止時，App Mesh 會自動設定用戶端以交涉 TLS，並針對憑證鏈結至的根使用者 CA 驗證憑證。
- 檔案型 TLS 憑證 — 當伺服器使用 Proxy 本機檔案系統的憑證設定 TLS 終止時，App Mesh 會自動設定用戶端以交涉 TLS，但不會驗證伺服器的憑證。

主題替代名稱

您可以選擇性地指定要信任的主體別名 (SAN) 清單。SAN 的格式必須為 FQDN 或 URI 格式。如果提供 SAN，Envoy 會驗證所顯示憑證的主體替代名稱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**
>
> 只有當 TLS 的用戶端原則設定為時，您才能使用萬用字元 SAN。`not enforced`如果用戶端虛擬節點或虛擬閘道的用戶端原則設定為強制執行 TLS，則無法接受萬用字元 SAN。

驗證加密

啟用 TLS 後，您可以查詢 Envoy 代理以確認通訊是否已加密。Envoy 代理伺服器會發出有關資源的統計資料，以協助您瞭解 TLS 通訊是否正常運作。例如，Envoy 代理伺服器會記錄其針對指定網狀端點交涉的成功 TLS 交涉次數的統計資料。判斷使用下列命令命名*my-mesh-endpoint*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，網狀端點有三個交握，因此通訊已加密。

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

當 TLS 協商失敗時，特使代理也會發出統計信息。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.*\(fail\|error
\)"
```

在傳回輸出的範例中，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 統計資料的相關資訊，請參閱使用者監聽程式統計資料。

憑證續約

AWS Private CA

當您使用 ACM 更新憑證時，續訂的憑證會在續訂完成後的 35 分鐘內自動發佈到連線的 Proxy。我們建議您使用受管理續約，以便在憑證有效期結束時自動續約憑證。有關更多信息，請參閱用戶指 AWS Certificate Manager 南中的 ACM 亞馬遜頒發的證書的管理續訂。

您自己的憑證

使用本機檔案系統中的憑證時，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 程序以載入新憑證。您也可以將較新的憑證置於不同的檔案路徑，並使用該檔案路徑更新虛擬節點或閘道組態。

將 Amazon ECS 工作負載設定為搭配使用 TLS 身份驗證 AWS App Mesh

您可以將網格設定為使用 TLS 驗證。請確定您新增至工作負載的 Envoy Proxy 側車可使用這些憑證。您可以將 EBS 或 EFS 磁碟區附加到您的特使附屬車上，也可以從 AWS Secrets Manager 儲存和擷取憑證。

- 如果您使用以檔案為基礎的憑證散發，請將 EBS 或 EFS 磁碟區附加至 Envoy 附屬程式。請確定憑證和私密金鑰的路徑與中設定的路徑相符 AWS App Mesh。
- 如果您使用的是基於 SDS 的分發，請添加一個實現 Envoy 的 SDS API 並可訪問證書的附屬軟件。

> **Note**
>
> Amazon ECS 不支援 SPIRT。

將 Kubernetes 工作負載設定為搭配使用 TLS 驗證 AWS App Mesh

您可以設定 Kubernetes 的 AWS App Mesh 控制器，以針對虛擬節點和虛擬閘道服務後端和接聽程式啟用 TLS 驗證。確保憑證可供您新增至工作負載的 Envoy Proxy 側車使用。您可以在相互 TLS 驗證的逐步解說一節中查看每種散佈類型的範例。

- 如果您使用以檔案為基礎的憑證散發，請將 EBS 或 EFS 磁碟區附加至 Envoy 附屬程式。請確定憑證和私密金鑰的路徑與控制器中設定的路徑相符。或者，您也可以使用掛載在檔案系統上的 Kubernetes 密碼。
- 如果您使用的是基於 SDS 的分發，則應設置實現 Envoy SDS API 的節點本地 SDS 提供程序。特使將通過 UDS 達到它。若要在 EKS AppMesh 控制器中啟用以 SDS 為基礎的 MTL 支援，請將旗標設定為，`true`並透過`enable-sds`旗標將本機 SDS 提供者的 UDS 路徑提供給控制器。`sds-uds-path`如果您使用 helm，請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 彈性 Kubernetes 服務（亞馬遜 EKS），您將無法使用 SPIRE 來分發證書。

交互 TLS 驗證

相互 TLS (傳輸層安全性) 驗證是 TLS 的選用元件，可提供雙向對等驗證。相互 TLS 驗證會透過 TLS 增加一層安全性，並允許您的服務驗證正在進行連線的用戶端。

用戶端與伺服器關係中的用戶端也會在工作階段交涉程序期間提供 X.509 憑證。伺服器會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受信任的憑證授權單位 (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 驗證 AWS App Mesh。它們是
TCP，HTTP /1.1，HTTP/2，gRPC。

> **Note**
>
> 使用應用程式網格，您可以為來自您服務的 Envoy 代理伺服器之間的通訊設定相互 TLS 驗證。不過，您的應用程式和 Envoy 代理伺服器之間的通訊是未加密的。

相互 TLS 驗證憑證

AWS App Mesh 支援兩個可能的憑證來源進行相互 TLS 驗證。TLS 用戶端原則中的用戶端憑證和監聽器 TLS 組態中的伺服器驗證可以來自：

- 檔案系統：來自正在執行之 Envoy Proxy 本機檔案系統的憑證。要將證書分發給 Envoy，您需要提供證書鏈的文件路徑和 App Mesh API 的私鑰。
- 特使的秘密發現服務（SDS）- 實施 SDS 並允許將證書發送給特使的 B ring-your-own 側車。它們包括 SPEFFE 運行時環境（SPIRE）。

> **Important**
>
> App Mesh 不會儲存用於相互 TLS 驗證的憑證或私密金鑰。相反，特使將它們存儲在內存中。

設定網面端點

為您的網狀端點 (例如虛擬節點或閘道) 設定相互 TLS 驗證。這些端點提供憑證並指定信任的授權單位。

若要這麼做，您需要為用戶端和伺服器佈建 X.509 憑證，並在 TLS 終止和 TLS 起始的驗證內容中明確定義受信任的授權單位憑證。

網格內部的信任

伺服器端憑證在虛擬節點接聽程式 (TLS 終止) 中設定，而用戶端憑證則在虛擬節點服務後端 (TLS 起始) 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原則，然後視需要為特定後端覆寫此原則。虛擬閘道只能使用套用至其所有後端的預設用戶端策略來設定。

您可以針對兩個網格的虛擬閘道上的輸入流量啟用相互 TLS 驗證，在不同網格之間設定信任。

網格之外的信任

在虛擬閘道接聽程式中指定用於 TLS 終止的伺服器端憑證。設定與虛擬閘道通訊的外部服務，以顯示用戶端憑證。憑證應衍生自伺服器端憑證在 TLS 產生的虛擬閘道接聽程式上使用的相同憑證授權單位 (CA) 之一。

將服務遷移至相互 TLS 驗證

將 App Mesh 中的現有服務移轉至相互 TLS 驗證時，請遵循這些準則來維持連線。

移轉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. 在您的伺服器上設定相互 TLS 驗證，指定伺服器憑證、信任鏈，以及選擇性地指定受信任的 SAN。
3. 確認通信正在通過 TLS 連接進行。
4. 在用戶端上設定相互 TLS 驗證，指定用戶端憑證、信任鏈，以及選擇性地指定受信任的 SAN。

5. STRICT啟用伺服器上 TLS 組態的模式。

移轉透過 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，並選擇性地指定受信任的 SAN。在後端伺服器要求之前，用戶端憑證不會傳送至後端。
2. 在伺服器上設定相互 TLS 設定，指定信任鏈，以及選擇性地指定受信任的 SAN。為此，您的服務器請求客戶端證書。

驗證相互 TLS 驗證

您可以參考傳輸層安全性：驗證加密文件，以瞭解 Envoy 如何確切發出 TLS 相關統計資料。對於相互 TLS 驗證，您應該檢查以下統計資料：

- `ssl.handshake`
- `ssl.no_certificate`
- `ssl.fail_verify_no_cert`
- `ssl.fail_verify_san`

下列兩個統計資料範例共同顯示終止至虛擬節點的 TLS 連線成功，全部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體別名 (SAN) 不符合用戶端信任的任何 SAN。

```
cluster.cds_egress_my-mesh_my-backend-node_http_9080.ssl.fail_verify_san: 5
```

App Mesh 相互 TLS 驗證逐步解說

- 相互 TLS 驗證逐步解說：本逐步解說說明如何使用 App Mesh CLI 建置具有相互 TLS 驗證的色彩應用程式。
- Amazon EKS 相互 TLS 軟體開發套件逐步解說：本逐步解說說明如何在 Amazon EKS 和 SPIFE 執行階段環境 (SPIRE) 中使用以 TLS SDK 為基礎的相互身份驗證。

- Amazon EKS 相互 TLS 檔案式逐步解說：本逐步解說說明如何在 Amazon EKS 和 SPIFE 執行階段環境 (SPIRE) 中使用以 TLS 檔案為基礎的相互身份驗證。

如何與 IAM AWS App Mesh 搭配使用

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以驗證 (登入) 和授權 (具有權限) 以使用 App Mesh 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

物件

根據您在 App Mesh 中執行的工作而定，使用方式 AWS Identity and Access Management (IAM) 會有所不同。

服務使用者 — 如果您使用 App Mesh 服務執行工作，則系統管理員會為您提供所需的認證和權限。當您使用更多 App Mesh 功能來完成工作時，您可能需要其他權限。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，請參閱疑難排解 AWS App Mesh 身分和存取。

服務管理員 — 如果您負責公司的 App Mesh 資源，您可能擁有對 App Mesh 的完整存取權。判斷您的服務使用者應存取哪些 App Mesh 功能和資源是您的工作。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，請參閱如何與 IAM AWS App Mesh 搭配使用。

IAM 管理員 — 如果您是 IAM 管理員，可能需要瞭解如何撰寫政策來管理 App Mesh 存取權的詳細資訊。若要檢視可在 IAM 中使用的應用程式 Mesh 身分型政策範例，請參閱。AWS App Mesh 以識別為基礎的原則範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴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識別的範例。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 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

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) 來增加帳戶的安全性。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 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

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，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 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 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，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，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要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 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。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。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。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 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

- 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。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 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。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 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。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暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，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哪些內容。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。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 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。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

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。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理

的策略。若要了解如何在受管政策及內嵌政策間選擇，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。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (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 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 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原則類型。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。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。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 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。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。如需更多資訊，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。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

如何與 IAM AWS App Mesh 搭配使用

在您使用 IAM 管理應用 App Mesh 的存取權限之前，您應該瞭解哪些 IAM 功能可用於應用 App Mesh。若要取得 App Mesh 和其他 AWS 服務如何與 IAM 搭配運作的高階檢視，請參閱 IAM 使用者指南中的與 IAM 搭配使用的AWS 服務。

主題

- App Mesh 身分識別型原則
- App Mesh 資源型原則
- 基於 App Mesh 標籤的授權
- App Mesh IAM 角色

App Mesh 身分識別型原則

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哪些內容。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。

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。原則動作通常與關聯的 AWS API 作業具有相同的名稱。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 中的原則動作會在動作之前使用下列前置詞：`appmesh:`。例如，若要授與某人在具有 `appmesh:ListMeshes` API 作業的帳戶中列出網格的權限，您可以將該`appmesh:ListMeshes`動作包含在他們的政策中。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*"
```

若要查看應用 App Mesh 動作清單，請參閱《IAM 使用者指南》 AWS App Mesh中的 「定義動作」。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。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": "*"
```

應用程式網格mesh資源具有以下 ARN。

```
arn:${Partition}:appmesh:${Region}:${Account}:mesh/${MeshName}
```

如需 ARN 格式的詳細資訊，請參閱 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。

例如，若要在陳述式中指定區域中的網格命名#用#式，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/*"
```

某些 App Mesh 動作 (例如用來建立資源的動作) 無法在特定資源上執行。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，使用虛擬節點目標CreateRoute建立路由，因此 IAM 使用者必須擁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 ARN 的清單，請參閱 IAM 使用者指南 AWS App Mesh中的定義資源。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 支援使用某些全域條件金鑰。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 AWS App Mesh中的條件金鑰。若要瞭解可搭配條件索引鍵使用哪些動作和資源，請參閱動作定義者 AWS App Mesh。

範例

若要檢視以 App Mesh 身分識別為基礎的原則範例，請參閱。AWS App Mesh 以識別為基礎的原則範例

App Mesh 資源型原則

App Mesh 不支援以資源為基礎的原則。

基於 App Mesh 標籤的授權

您可以將標籤附加至 App Mesh 資源，或將要求中的標籤傳遞至 App Mesh。若要根據標籤控制存取，請使用 appmesh:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件金鑰，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，請參閱使用受限標籤建立應用程式網格。

App Mesh IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的實體。

搭配應用 App Mesh 使用臨時憑

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或等 AWS STS API 作業來取得臨時安全登入資料GetFederationToken。

App Mesh 支援使用臨時認證。

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。如需有關建立或管理 App Mesh 服務連結角色的詳細資訊，請參閱對應用 App Mesh 使用服務連結角色。

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 以識別為基礎的原則範例

根據預設，IAM 使用者和角色沒有建立或修改 App Mesh 資源的權限。他們也無法使用 AWS Management Console AWS CLI、或 AWS API 執行工作。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

- 使用應用程式網格主控台
- 允許使用者檢視他們自己的許可
- 建立網面
- 列出並描述所有網格
- 使用受限標籤建立應用程式網格

政策最佳實務

以身分識別為基礎的原則會決定某人是否可以建立、存取或刪除您帳戶中的 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 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。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。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。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。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，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，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 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。

使用應用程式網格主控台

若要存取 AWS App Mesh 主控台，您必須擁有最少一組權限。這些權限必須允許您列出並檢視 AWS 帳戶中 App Mesh 資源的詳細資料。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受管理的策略附加到使用者。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。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。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。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "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": "*"
        }
    ]
```

```
}
```

建立網面

此範例顯示如何建立策略，以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

此範例顯示如何建立允許使用者以唯讀方式存取清單和描述所有網格的原則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appmesh:DescribeMesh",
                "appmesh:ListMeshes"
            ],
            "Resource": "*"
        }
    ]
}
```

使用受限標籤建立應用程式網格

您可以在 IAM 政策中使用標籤來控制哪些標籤可以在 IAM 請求中傳遞。您可以指定哪些標籤鍵值配對可以從 IAM 使用者或角色新增、變更或移除。此範例顯示如何建立允許建立網狀的政策，但前提是使用名為 *TeamName* 的標籤和 *Book* Steam 值建立網格。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:CreateMesh",
            "Resource": "*",
            "Condition": {
                "ForAnyValue: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面，網面必須包含名為的標籤teamName和值booksTeam。如果網面不包含此標籤和值，則建立網面的嘗試會失敗。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。

AWSApp Mesh 的受管原則

AWS 受管政策是由 AWS 建立和管理的獨立政策。AWS 受管政策的設計在於為許多常見使用案例提供許可，如此您就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與您特定使用案例的最低權限許可，因為它們可供所有 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。

您無法更改 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，更新會影響政策連接的所有主體身分 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies中的 AWS 受管政策。

AWS 受管政策： AWSAppMeshServiceRolePolicy

您可以將 AWSAppMeshServiceRolePolicy 連接到 IAM 實體。允許存取使用或管理的AWS服務和資源AWS App Mesh。

若要檢視此原則的權限，請參閱AWS受管理AWSAppMeshServiceRolePolicy的策略參考中的。

如需有關的權限詳細資訊AWSAppMeshServiceRolePolicy，請參閱 App Mesh 的服務連結角色權限。

AWS 受管政策： AWSAppMeshEnvoyAccess

您可以將 AWSAppMeshEnvoyAccess 連接到 IAM 實體。用於訪問虛擬節點配置的應用 Mesh 特許政策。

若要檢視此原則的權限，請參閱AWS受管理AWSAppMeshEnvoyAccess的策略參考中的。

AWS 受管政策： AWSAppMeshFullAccess

您可以將 AWSAppMeshFullAccess 連接到 IAM 實體。提供對 AWS App Mesh API 和AWS Management Console.

若要檢視此原則的權限，請參閱AWS受管理AWSAppMeshFullAccess的策略參考中的。

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 Mesh 預覽特使政策。

若要檢視此原則的權限，請參閱AWS受管理AWSAppMeshPreviewEnvoyAccess的策略參考中的。

AWS 受管政策： AWSAppMeshPreviewServiceRolePolicy

您可以將 AWSAppMeshPreviewServiceRolePolicy 連接到 IAM 實體。允許存取使用或管理的AWS服務和資源AWS App Mesh。

若要檢視此原則的權限，請參閱AWS受管理AWSAppMeshPreviewServiceRolePolicy的策略參考中的。

AWS 受管政策： AWSAppMeshReadOnly

您可以將 AWSAppMeshReadOnly 連接到 IAM 實體。提供 AWS App Mesh API 和AWS Management Console.

若要檢視此原則的權限，請參閱AWS受管理AWSAppMeshReadOnly的策略參考中的。

AWS 受管政策的 AWS App Mesh 更新項目

檢視自 AWS App Mesh 開始追蹤 AWS 受管政策變更以來的更新詳細資訊。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
AWSAppMeshServiceRolePolicy, AWSServiceRoleForAppMesh— 更新的政策。	已更AWSAppMeshServiceRolePolicy 新AWSServiceRoleForAppMesh 並允許存取 AWS Cloud Map DiscoverInstancesRevision API。	2023年10月12日

若要提供存取權，請新增許可到您的使用者、群組或角色：

- AWS IAM Identity Center 中的使用者和群組：

  建立許可集合。請遵循《AWS IAM Identity Center 使用者指南》的建立許可集合中的指示。

- 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。請遵循《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。請遵循《IAM 使用者指南》的為 IAM 使用者建立角色中的指示。
  - (不建議) 將政策直接連接至使用者，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。

對應用 App Mesh 使用服務連結角色

AWS App Mesh 使用 AWS Identity and Access Management (IAM) 服務連結的角色。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，並包含服務代表您呼叫其他服AWS務所需的所有權限。

服務連結角色可讓您更輕鬆地設定 App Mesh，因為您不需要手動新增必要的權限。App Mesh 會定義其服務連結角色的權限，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，並且該許可政策不能連接到任何其他 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。這可以保護您的 App Mesh 資源，因為您無法不小心移除存取資源的權限。

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 Yes (是)，以檢視該服務的服務連結角色文件。

App Mesh 的服務連結角色權限

App Mesh 使用名為的服務連結角色 AWSServiceRoleForAppMesh— 角色允許 App Mesh 代表您呼叫AWS服務。

服 AWSServiceRoleForAppMesh 務連結角色會信任`appmesh.amazonaws.com`服務擔任該角色。

許可權詳細

- `servicediscovery:DiscoverInstances`-允許 App Mesh 對所有AWS資源完成操作。
- `servicediscovery:DiscoverInstancesRevision`-允許 App Mesh 對所有AWS資源完成操作。

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。

建立 App Mesh 的服務連結角色

如果您在 2019 年 6 月 5 日之後在、或 AWS API 中AWS Management Console建立網格AWS CLI，App Mesh 會為您建立服務連結角色。若要為您建立服務連結角色，您用來建立網格的 IAM 帳戶必須已附加 AWSAppMeshFullAccessIAM 政策，或附加包含iam:CreateServiceLinkedRole權限的政策。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。如果您的帳戶僅包含 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用於這些網格，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 CLI 或 AWS API 中，建立一個服務名稱為 appmesh.amazonaws.com 的服務連結角色。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 服務連結的角色。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**
>
> 當您嘗試刪除資源時，如果 App Mesh 服務正在使用該角色，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。

若要刪除使用的應用程式網格資源 AWSServiceRoleForAppMesh

1. 刪除網格中為所有路由器定義的所有路由。
2. 刪除網狀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。

對帳戶中的所有網格完成上述步驟。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForAppMesh 服務連結角色。AWS CLI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 服務連結角色的支援區域

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 App Mesh 端點和配額。

特使代理授權

代理授權授權授權在 Amazon ECS 任務內執行的特使代理、在 Amazon EKS 上執行的 Kubernetes 網繭中執行，或在 Amazon EC2 執行個體上執行的執行個體中，從 App Mesh Envoy 管理服務讀取一或多個網狀端點的組態。對於在 2021 年 4 月 26 日之前已經擁有 Envoys 連線至其 App Mesh 端點的客戶帳戶，使用傳輸層安全性 (TLS) 的虛擬節點和虛擬閘道 (有無 TLS) 的虛擬節點需要 Proxy 授權。對於想要在 2021 年 4 月 26 日之後將 Envoys 連線到其 App Mesh 端點的客戶帳戶，所有 App Mesh 功能都需要代理授權。建議所有客戶帳戶對所有虛擬節點啟用 Proxy 授權，即使他們不使用 TLS，以獲得使用 IAM 授權特定資源的安全且一致的體驗。代理授權要求在 IAM 政策中指定權`appmesh:StreamAggregatedResources`限。該政策必須附加到 IAM 角色，並且該 IAM 角色必須附加到託管代理的計算資源。

建立 IAM 政策

如果您希望服務網格中的所有網格端點都能夠讀取所有網格端點的組態，請跳至建立 IAM 角色。如果要限制個別網狀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。建議將可從中讀取組態的網狀端點限制為僅在特定計算資源上執行的 Envoy Proxy。建立 IAM 政策並將`appmesh:StreamAggregatedResources`許可新增至政策中。下列範例原則允許在服務網格中讀取名`serviceBv2`為`serviceBv1`和的虛擬節點的組態。無法讀取服務網格中定義的任何其他虛擬節點的組態。如需有關建立或編輯 IAM 政策的詳細資訊，請參閱建立 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"
            ]
        }
    ]
}
```

您可以建立多個策略，每個策略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狀端點都能夠讀取所有網狀端點的組態，則只需建立一個 IAM 角色即可。如果您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策略建立角色。完成 Proxy 執行所在之計算資源的指示。

- Amazon EKS — 如果您想要使用單一角色，則可以使用在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為服務帳戶啟用 IAM 角色中定義的要求。建立身分與存取權管理角色，並將這些角色與 Kubernetes 服務帳戶建立關聯。如需詳細資訊，請參閱為服務帳戶建立 IAM 角色和政策和為服務帳戶指定 IAM 角色。
- Amazon ECS — 選取AWS服務、選取彈性容器服務，然後在建立 IAM 角色時選取彈性容器服務任務使用案例。
- 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。無論您是直接在 Amazon EC2 執行個體上託管代理伺服器，或是在執行個體上執行個體執行個體上執行個體上執行個體或

如需有關建立 IAM 角色的詳細資訊，請參閱建立 IAM 角色，請參閱建立 IAMAWS

連接 IAM 政策

如果您希望服務網格中的所有網狀端點都能讀取所有網狀端點的組態，請將AWSAppMeshEnvoyAccess受管 IAM 政策附加到您在上一個步驟中建立的 IAM 角色。如果您要限

制個別網格端點可從中讀取組態的網格端點，請將您建立的每個策略附加到您建立的每個角色。如需將自訂或受管 IAM 政策附加至 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 角色附加到適當的運算資源：

- Amazon EKS — 如果您將政策附加到工作者節點上的角色，則可以略過此步驟。如果您建立不同的角色，請將每個角色指派給個別的 Kubernetes 服務帳戶，並將每個服務帳戶指派給包含特使代理的個別 Kubernetes 網繭部署規格。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 角色和 Kubernetes 說明文件中的設定網繭的服務帳戶。
- 亞馬遜 ECS — 將 Amazon ECS 任務角色附加到包含特使代理的任務定義。可使用 EC2 或 Fargate 啟動類型來部署此任務。有關如何建立 Amazon ECS 任務角色並將其附加到任務的詳細資訊，請參閱為任務指定 IAM 角色。
- Amazon EC2 — IAM 角色必須附加到託管特使代理的 Amazon EC2 實例。有關如何將角色附加到 Amazon EC2 執行個體的詳細資訊，請參閱我已建立 IAM 角色，現在我想將其指派給 EC2 執行個體。

確認許可

選取其中一個計算服務名稱，確認已將`appmesh:StreamAggregatedResources`權限指派給託管 Proxy 的計算資源。

Amazon EKS

可將自訂原則指派給指派給工作者節點的角色、個別網繭或兩者。不過，建議您僅在個別網繭上指派原則，以便將個別網繭的存取限制為個別網狀端點。如果政策附加到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此處找到的步驟。若要判斷指派給 Kubernetes 網繭的身分與存取權管理角色，請完成下列步驟。

1. 檢視 Kubernetes 部署的詳細資料，其中包含您要確認已指派 Kubernetes 服務帳戶的網繭。下列命令會檢視名為 *my-deployment* 部署的詳細資料。

   ```
   kubectl describe deployment my-deployment
   ```

   在返回的輸出中，請注意右側的值`Service Account:`。如果開頭為的行`Service Account:`不存在，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. 檢視上一個步驟傳回之服務帳戶的詳細資訊。下列命令會檢視名為的服務帳戶的詳細資料*my-service-account*。

```
kubectl describe serviceaccount my-service-account
```

假設 Kubernetes 服務帳戶與AWS Identity and Access Management角色相關聯，則傳回的其中一行看起來會類似下列範例。

```
Annotations:         eks.amazonaws.com/role-arn=arn:aws:iam::123456789012:role/
my-deployment
```

在上一個範例中，`my-deployment`是與服務帳戶相關聯的 IAM 角色名稱。如果服務帳戶輸出未包含與上述範例類似的行，則 Kubernetes 服務帳戶不會與帳戶相關聯，因此您需要將其關聯至AWS Identity and Access Management帳戶。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，並開啟位於 https://console.aws.amazon.com/iam/ 的 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. 確認已列出您先前建立的自訂原則或受`AWSAppMeshEnvoyAccess`管理的原則。如果兩個政策都未附加，請將 IAM 政策附加到 IAM 角色。如果您想要附加自訂 IAM 政策，但沒有自訂 IAM 政策，則需要建立具有所需許可的自訂 IAM 政策。如果已附加自訂 IAM 政策，請選取該政策並確認其包含`"Action": "appmesh:StreamAggregatedResources"`。如果沒有，則您需要將該權限新增至自訂 IAM 政策。您也可以確認列出特定的網狀端點適當的 Amazon Resource Name (ARN)。如果未列出任何 ARN，則您可以編輯策略以新增、移除或變更列出的 ARN。如需詳細資訊，請參閱編輯 IAM 政策和建立 IAM 政策.
6. 針對包含特使代理伺服器的每個 Kubernetes 網繭重複上述步驟。

Amazon ECS

1. 在 Amazon ECS 主控台中，選擇「任務定義」。
2. 選擇您的 ECS S S S S S S S S S
3. 在「作業定義名稱」頁面上，選取您的作業定義。
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附加到您的任務。

5. 在 [摘要] 頁面的 [權限] 索引標籤上，確認已列出您先前建立的自訂原則或AWSAppMeshEnvoyAccess受管理的原則。如果兩個政策都未附加，請將 IAM 政策附加到 IAM 角色。如果您想要附加自訂 IAM 政策，但沒有自訂 IAM 政策，則需要建立自訂 IAM 政策。如果已附加自訂 IAM 政策，請選取該政策並確認其包含"Action": "appmesh:StreamAggregatedResources"。如果沒有，則您需要將該權限新增至自訂 IAM 政策。您也可以確認列出適當的 Amazon Resource Name (ARN)。如果未列出任何 ARN，則您可以編輯策略以新增、移除或變更列出的 ARN。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 代理的每個工作定義重複上述步驟。

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 代理伺服器的其中一個執行個體。
3. 在 [說明] 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 [權限] 索引標籤上，確認已列出您先前建立的自訂原則或AWSAppMeshEnvoyAccess受管理的原則。如果兩個政策都未附加，請將 IAM 政策附加到 IAM 角色。如果您想要附加自訂 IAM 政策，但沒有自訂 IAM 政策，則需要建立自訂 IAM 政策。如果已附加自訂 IAM 政策，請選取該政策並確認其包含"Action": "appmesh:StreamAggregatedResources"。如果沒有，則您需要將該權限新增至自訂 IAM 政策。您也可以確認列出適當的 Amazon Resource Name (ARN)。如果未列出任何 ARN，則您可以編輯策略以新增、移除或變更列出的 ARN。如需詳細資訊，請參閱編輯 IAM 政策和建立 IAM 政策.
5. 針對託管 Envoy 代理伺服器的每個執行個體重複上述步驟。

疑難排解 AWS App Mesh 身分和存取

使用下列資訊可協助您診斷並修正使用 App Mesh 和 IAM 時可能會遇到的常見問題。

主題

- 我沒有授權在 App Mesh 中執行動作
- 我想要允許 AWS 帳戶以外的人員存取我的 App Mesh 資源

我沒有授權在 App Mesh 中執行動作

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。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建立名為 *my-mesh* 的網格*my-virtual-node*中的虛擬節點，但沒有appmesh:CreateVirtualNode權限時，就會發生下列錯誤。

```
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原則，以允許他使用此appmesh:CreateVirtualNode動作建立虛擬節點。

> **Note**
>
> 由於虛擬節點是在網格內建立的，因此 Mateo 的帳號也需要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 帳戶以外的人員存取我的 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：

- 若要瞭解 App Mesh 是否支援這些功能，請參閱如何與 IAM AWS App Mesh 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱 IAM 使用者指南中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源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第三方 AWS 帳戶 擁有的存取權。
- 若要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)。
- 若要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 角色與資源型政策的差異。

使用記錄 AWS App Mesh API 呼叫 AWS CloudTrail

AWS App Mesh 與提供使用者AWS CloudTrail、角色或使用者所採取之動作記錄的服務整合 AWS 服務。 CloudTrail 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，以及對應用程式 Mesh API 作業的程式碼呼叫。使用收集的資訊 CloudTrail，您可以判斷向 App Mesh 提出的要求、提出要求的來源 IP 位址、提出要求的時間，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 身分中心使用者提出。
- 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。

CloudTrail 在您創建帳戶 AWS 帳戶 時處於活動狀態，並且您自動可以訪問 CloudTrail 事件歷史記錄。 CloudTrail 事件歷史記錄提供了過去 90 天中記錄的管理事件的可查看，可搜索，可下載和不可變的記錄。 AWS 區域若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 事件歷程〉。查看活動歷史記錄不 CloudTrail收取任何費用。

如需過 AWS 帳戶 去 90 天內持續的事件記錄，請建立追蹤或 CloudTrailLake 事件資料存放區。

CloudTrail 小徑

追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。使用建立的所有系統線 AWS Management Console 都是多區域。您可以使用建立單一區域或多區域系統線。 AWS CLI建議您建立多區域追蹤，因為您會擷取帳戶 AWS 區域 中的所有活動。如果您建立單一區域追蹤，則只能檢視追蹤記錄中的 AWS 區域事件。如需有關追蹤的詳細資訊，請參閱《AWS CloudTrail 使用指南》中的「為您的建立追蹤」 AWS 帳戶和「為組織建立追蹤」。

您可以透 CloudTrail 過建立追蹤，免費將一份正在進行的管理事件副本傳遞到 Amazon S3 儲存貯體，但是需要支付 Amazon S3 儲存費用。如需有關 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 湖泊事件資料存放區

CloudTrail L@@ ake 可讓您針對事件執行 SQL 型查詢。 CloudTrail 湖將基於行的 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。若要取得有關 CloudTrail Lake 的更多資訊，請參閱使用指南中的〈AWS CloudTrail 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。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。

App Mesh 管理事件 CloudTrail

管理事件提供有關在您的資源上執行的管理作業的資訊 AWS 帳戶。這些也稱為控制平面操作。依預設，會 CloudTrail 記錄管理事件。

AWS App Mesh 將所有 App Mesh 控制平面作業記錄為管理事件。如需 App Mesh 記錄到的 AWS App Mesh 控制平面作業清單 CloudTrail，請參閱 AWS App Mesh API 參考資料。

App Mesh 事件範

事件代表來自任何來源的單一請求，並包括有關請求的 API 操作，操作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此事件不會以任何特定順序顯示。

下列範例顯示示範`StreamAggregatedResources`動作的 CloudTrail 記錄項目。

```
{
    "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"
}
```

若要取得有關 CloudTrail 記錄內容的資訊，請參閱AWS CloudTrail 使用指南中的CloudTrail記錄內容。

AWS App Mesh 中的資料保護

AWS 共同的責任模型適用於 AWS App Mesh 中的資料保護。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您負責維護在此基礎設施上託管內容的控制權。您也必須負責您所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，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，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，並設定個人使用者。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 AWS CloudTrail 設定 API 和使用者活動日誌記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 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。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，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) 140-2 概觀。

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 Name (名稱) 欄位。這包括當您使用主控台、API 或 AWS SDK AWS 服務 使用 App Mesh 或其他應用程式網格時。AWS CLI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

使用應用程式網格時，您的資料會加密。

靜態加密

您建立的所有資料和設定都會在靜態時加密。

傳輸中加密

App Mesh 端點使用 HTTPS 通訊協定。特使代理伺服器與 App Mesh 特使管理服務之間的所有通訊均經過加密。虛擬節點內容器之間的通訊並未加密，但此流量不會離開網路命名空間。

AWS App Mesh 的合規驗證

要瞭解 AWS 服務 是否在特定法規遵循方案範圍內，請參閱法規遵循方案範圍內的 AWS 服務，並選擇您感興趣的法規遵循方案。如需一般資訊，請參閱 AWS 法規遵循方案。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱 AWS Artifact 中的下載報告。

您使用 AWS 服務 時的法規遵循責任取決於資料的敏感度、您的公司的合規目標，以及適用的法律和法規。AWS 提供以下資源協助您處理法規遵循事宜：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS 上部署以安全及合規為重心的基準環境的架構考量和步驟。
- Amazon Web Services 的 HIPAA 安全與法規遵循架構：本白皮書說明公司可如何運用 AWS 來建立符合 HIPAA 規定的應用程式。

> **Note**
>
> 並非全部的 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源：這組手冊和指南可能適用於您的產業和位置。

- AWS 客戶合規指南：透過合規的角度瞭解共同的責任模式。這份指南橫跨多個架構 (包含國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準組織 (ISO))，總結保護 AWS 服務的最佳實務並將指導方針對應至安全控制。
- AWS Config 開發人員指南中的使用規則評估資源：AWS Config 服務可評估您的資源組態對於內部實務、業界準則和法規的合規狀態。
- AWS Security Hub – 此 AWS 服務 可供您全面檢視 AWS 中的安全狀態。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 controls reference。
- AWS Audit Manager – 此 AWS 服務 可協助您持續稽核 AWS 使用情況，以簡化管理風險與法規與業界標準的法規遵循方式。

基礎架構安全性 AWS App Mesh

作為託管服務， AWS App Mesh 受到 AWS 全球網絡安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取應用程式網格。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。

您可以將應用程式網格設定為使用介面 VPC 端點，以改善 VPC 的安全性狀態。如需更多詳細資訊，請參閱 App Mesh 介面 VPC 端點 ()AWS PrivateLink。

App Mesh 介面 VPC 端點 ()AWS PrivateLink

您可以將應用程式網格設定為使用介面 VPC 端點，以改善 Amazon VPC 的安全狀態。介面端點採用這項技術 AWS PrivateLink，可讓您使用私有 IP 位址私有存取 App Mesh API。 PrivateLink將 Amazon VPC 和應用程式網格之間的所有網路流量限制在 Amazon 網路。

您不需要進行設定 PrivateLink，但我們建議您這麼做。如需 VPC 端點 PrivateLink 和連接 VPC 端點的詳細資訊，請參閱透過 AWS PrivateLink存取服務。

應用程式網格介面 VPC 端點的注意事項

在為 App Mesh 設定介面 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，而且您的任務使用日awslogs誌驅動程式將日誌資訊傳送到 CloudWatch 日誌，則必須為 CloudWatch 日誌建立介面 VPC 端點。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用指南中的將日 CloudWatch 誌與界面 VPC 端點搭配使用。
- VPC 端點不支援 AWS 跨區域要求。確保您在計劃向 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來自 Amazon VPC 私有子網路的連入連線，連接埠 443 上的連入連線。

> **Note**
>
> Envoy 連線不支援透過將端點原則附加到 VPC 端點 (例如，使用服務名稱com.amazonaws.*Region*.appmesh-envoy-management) 來控制對應用程式網格的存取。

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內容與限制。

為應用程式網格建立介面 VPC 端點

若要為應用 App Mesh 服務建立介面 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，以連接到 App Mesh 的公共特使管理服務以及用com.amazonaws.*Region*.appmesh於網格操作。

> **Note**
>
> *##*代表 App Mesh 支援的 AWS 區域識別碼，us-east-2例如美國東部 (俄亥俄) 區域。

雖然您可以在任何支援 App Mesh 的區域中為 App Mesh 定義介面 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。若要瞭解某個區域中的介面 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，下列命令會傳回可在美國東部 (俄亥俄) 區域內部署 App Mesh 介面 VPC 端點的可用區域：

```
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[]'
```

AWS App Mesh 中的恢復能力

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，它們以低延遲、高輸送量和高度備援聯網功能相互連結。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 可用性。App Mesh 會自動偵測並取代有問題的控制平面執行個體，而且提供自動化版本升級及修補。

災難恢復AWS App Mesh

App Mesh 服務會管理客戶資料的備份。您無需執行任何操作即可管理備份。備份資料已加密。

中的配置和漏洞分析 AWS App Mesh

App Mesh 會出售託管的 Envoy 代理 Docker 容器映像，您可以使用您的微服務部署。App Mesh 可確保容器映像檔已修補最新的弱點和效能修補程式。在向您提供圖像之前，應用程序 Mesh 會根據應用程序網格功能集測試新的特使代理版本。

您必須更新您的微服務才能使用更新的容器映像版本。以下是最新版本的映像檔。

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

App Mesh 疑

本章討論疑難排解最佳做法，以及使用 App Mesh 時可能會遇到的常見問題。選取下列其中一個領域，以檢閱該領域的最佳作法和常見問題。

主題

App Mesh 疑難排解最佳做法

建議您遵循本主題中的最佳做法，以疑難排解使用 App Mesh 時的問題。

啟用特使代理管理介面

Envoy Proxy 隨附一個管理介面，您可以使用該介面來探索設定和統計資料，以及執行其他管理功能，例如連線排空。如需詳細資訊，請參閱 Envoy 文件中的管理介面。

如果您使用受管理特使形象，則依預設會在連接埠 9901 上啟用管理端點。中提供的範例會將管理端點 URL 範例App Mesh 設定疑顯示為`http://my-app.default.svc.cluster.local:9901/`。

> **Note**
>
> 管理端點絕對不應暴露在公用網際網路上。此外，我們建議監視管理端點記錄，這些記錄檔由`ENVOY_ADMIN_ACCESS_LOG_FILE`環境變數預設設定為`/tmp/envoy_admin_access.log`。

為量度卸載啟用特使 DogStats D 整合

Envoy Proxy 可設定為卸載 OSI 第 4 層和第 7 層流量的統計資料，以及內部處理序健全狀況。本主題說明如何使用這些統計資料，而不卸載指標以接收量度 (例如 CloudWatch 指標和 Prometheus)，但將這些統計資料放在所有應用程式的集中位置，可協助您更快速診斷問題並確認行為。如需詳細資訊，請參閱使用 Amazon CloudWatch 指標和 Promethe us 文件。

您可以透過設定中定義的參數來設定 DogStats D 度量DogStatsD 型變數。如需 DogStats D 的詳細資訊，請參閱 DogStatsD 文件。您可以在使用 Amazon ECS 的應用程式網格中找到AWS CloudWatch 指標卸載的示範。 GitHub

啟用存取日誌

我們建議您在和上啟用存取記錄虛擬節點，虛擬閘道以探索應用程式之間傳輸流量的詳細資料。如需詳細資訊，請參閱 Envoy 文件中的存取記錄。記錄檔提供 OSI 第 4 層和第 7 層流量行為的詳細資訊。使用 Envoy 的預設格式時，您可以使用下列剖析陳述式使用日誌深入分析來分析存取CloudWatch記錄。

```
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

我們建議將 Envoy 代理的日誌級別設置為debug在生產前環境中。偵錯記錄可協助您識別問題，然後再將相關的 App Mesh 組態轉移到生產環境。

如果您使用的是 Envoy 映像檔，則可以debug透過ENVOY_LOG_LEVEL環境變數將記錄層級設定為。

> **Note**
>
> 我們不建議在生產環境中使用該debug級別。將層級設定為debug增加記錄，並可能影響效能和卸載至記錄檔等解決方案的整體記CloudWatch 錄成本。

當您使用 Envoy 的預設格式時，您可以使用下列剖析陳述式，利用日CloudWatch 誌深入分析來分析程序記錄：

```
parse @message "[*][*][*][*] [*] *" as Time, Thread, Level, Name, Source, Message
```

使 App Mesh 控制平面監控特使代理連線

我們建議您監視 Envoy 指標，`control_plane.connected_state`以確保 Envoy 代理與 App Mesh 控制平面進行通信，以獲取動態配置資源。如需詳細資訊，請參閱管理伺服器。

App Mesh 設定疑

本主題詳細說明您在使用 App Mesh 設定時可能會遇到的常見問題。

無法提取特使容器映像

徵狀

您在 Amazon ECS 任務中收到下列錯誤訊息。下列訊息中的 Amazon ECR *## ID* 和*##*可能會有所不同，具體取決於您從哪個 Amazon ECR 儲存庫中提取容器映像。

```
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'
```

解析度

此錯誤表示使用的任務執行角色沒有與 Amazon ECR 通訊的權限，也無法從存放庫中提取 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"
}
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法連接到 App Mesh 特使管理服務

徵狀

您的特使代理無法連接到 App Mesh 特使管理服務。您正在看到：

- 連線拒絕錯誤
- 連線逾時
- 解決應用程式網狀特使管理服務端點時發生錯誤
- gRPC 錯誤

解析度

請確定您的 Envoy Proxy 可存取網際網路或私有 VPC 端點，而且您的安全群組允許連接埠 443 上的輸出流量。App Mesh 的公共特使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
```

您可以使用下面的命令對 EMS 的連接進行調試。這會將有效但空白的 gRPC 請求傳送給特使管理服務。

```
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，表示您與 Envoy 管理服務的連線正常運作。有關對 gRPC 相關錯誤的調試，請參閱 Envoy 中斷與 App Mesh 特使管理服務斷開連接的錯誤，並顯示錯誤文本。

```
grpc-status: 16
grpc-message: Missing Authentication Token
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

特使與 App Mesh 特使管理服務斷開連接，並顯示錯誤文本

徵狀

您的特使代理無法連接到 App Mesh Envoy 管理服務並接收其配置。您的 Envoy 代理日誌包含如下所示的日誌條目。

```
gRPC config stream closed: gRPC status code, message
```

解析度

在大多數情況下，記錄檔的訊息部分應指出問題。下表列出您可能會看到的最常見的 gRPC 狀態碼、原因及解析度。

gRPC 狀態碼	原因	解析度
0	優雅地斷開特使管理服務。	沒有問題。App Mesh 偶爾斷開特使代理與此狀態代碼的連接。特使將重新連接並繼續接收更新。
3	找不到網狀端點 (虛擬節點或虛擬閘道) 或其關聯資源之一。	仔細檢查您的 Envoy 配置，以確保它具有它所代表的 App Mesh 資源的適當名稱。如果您的 App Mesh 資源與其他 AWS資源 (例如AWS Cloud Map命名空間或 ACM 憑證) 整合，請確定這些資源存在。
7	Envoy 代理伺服器未經授權執行動作，例如連線至 Envoy 管理服務或擷取相關資源。	請確定您建立的 IAM 政策具有適用於 App Mesh 和其他服務的政策陳述式，並將該政策附加到 Envoy Proxy 用於連線至 Envoy 管理服務的 IAM 使用者或角色。
8	指定 App Mesh 資源的 Envoy 代理數量超過帳戶層級服務配額。	如App Mesh需預設帳戶配額以及如何要求提高配額的資訊，請參閱。

gRPC 狀態碼	原因	解析度
16	Envoy 代理伺服器沒有的有效驗證憑證AWS。	確保 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 程序使用超過1024檔案描述元，則在 Envoy 版本v1.24和之前無法擷取認證的已知問題 #24136。當特使提供高流量時，就會發生這種情況。您可以在偵錯層級檢查文字「A libcurl function was given a bad argument」的 Envoy 記錄檔，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 洞察，觀察來自您的特使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，或者您的問題仍未解決，請考慮開啟GitHub 問題。

特使集裝箱健康檢查、就緒探測器或活力探測失敗

徵狀

您的特使代理伺服器在 Amazon ECS 任務、Amazon EC2 執行個體或 Kubernetes 網繭中的運作狀態檢查失敗。例如，您可以使用下列命令查詢 Envoy 管理介面，並接收以外LIVE的狀態。

```
curl -s http://my-app.default.svc.cluster.local:9901/server_info | jq '.state'
```

解析度

以下是根據 Envoy 代理傳回的狀態而定的補救步驟清單。

- PRE_INITIALIZING或者 INITIALIZING — 特使代理尚未接收配置，或者無法從 App Mesh Envoy 管理服務連接和檢索配置。嘗試連線時，特使可能會收到 Envoy 管理服務的錯誤訊息。如需詳細資訊，請參閱中的錯誤特使與 App Mesh 特使管理服務斷開連接，並顯示錯誤文本。
- DRAINING— Envoy 代理已開始排除連線，以回應 Envoy 管理介面上的/healthcheck/fail或/drain_listeners要求。除非您即將終止 Amazon ECS 任務、Amazon EC2 執行個體或 Kubernetes 網繭，否則我們不建議您在管理界面上叫用這些路徑。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

從負載平衡器到網狀端點的 Health 狀態檢查失敗

徵狀

容器健康狀態檢查或整備程度探查會將您的網狀端點視為狀況良好，但是從負載平衡器到網狀端點的健康狀態檢查失敗。

解析度

若要解決此問題，請完成下列工作。

- 確保與您的網狀端點關聯的安全群組接受您為健康狀態檢查設定的連接埠上的輸入流量。
- 確保手動要求時運作狀態檢查一致成功；例如，來自 VPC 內的防禦主機。
- 如果您要為虛擬節點設定健康狀態檢查，建議您在應用程式中實作健康狀態檢查端點，例如 HTTP 的 /ping。這樣可確保 Envoy Proxy 和應用程式都可從負載平衡器路由。
- 您可以針對虛擬節點使用任何彈性負載平衡器類型，視您需要的功能而定。如需詳細資訊，請參閱 Elastic Load Balancing 功能。
- 如果您要設定虛擬閘道的健全狀況檢查，建議您在虛擬閘道的接聽程式連接埠上使用具有 TCP 或 TLS 健康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啟動載入並準備好接受連線。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

虛擬閘道在 1024 或更少的連接埠上不接受流量

徵狀

您的虛擬閘道不接受連接埠 1024 以下的流量，但會接受大於 1024 的連接埠號碼上的流量。例如，您使用下列命令查詢 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"
```

您可能會在記錄檔中看到類似下列文字的文字，說明無法繫結至授權通訊埠：

```
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
```

解析度

若要解決此問題，指定給閘道的使用者必須具備 linux 功能CAP_NET_BIND_SERVICE。如需詳細資訊，請參閱《Linux 程式設計人員手冊》中的功能、ECS 工作定義參數中的 Linux 參數和 Kubernetes 文件中的 [設定容器的功能]。

> **Important**
>
> Fargate 必須使用大於 1024 的端口值。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

App Mesh 連線疑

本主題詳細說明您使用 App Mesh 連線時可能會遇到的常見問題。

無法解析虛擬服務的 DNS 名稱

徵狀

您的應用程式無法解析嘗試連線的虛擬服務的 DNS 名稱。

解析度

這是已知問題。如需詳細資訊，請參閱任何主機 VirtualServices 名稱或 FQDN 問題 GitHub 的名稱。App Mesh 中的虛擬服務可以命名為任何內容。只要虛擬服務名稱有 DNS A 記錄，且應用程式可以解析虛擬服務名稱，Envoy 就會代理要求並路由到適當的目的地。若要解決此問題，請將 DNS A 記錄新增至任何非迴路 IP 位址，例如10.10.10.10虛擬服務名稱。您可以在下列情況下新增 DNS A 記錄：

- 在 Amazon Route 53 中，如果名稱後綴為您的私有託管區域名稱
- 在應用程序容器的`/etc/hosts`文件中
- 在您管理的第三方 DNS 伺服器中

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

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，或者從應用程式的觀點來看要求可能會失敗，並顯示`HTTP 503`回應碼。

解析度

如果應用程式完全無法連線 (未傳回任何回`HTTP 503`應碼)，請執行下列動作：

- 請確定您的計算環境已設定為與 App Mesh 搭配使用。
  - 對於 Amazon ECS，請確定已啟用適當的代理伺服器組態。如需 end-to-end 逐步解說，請參閱應用程式網格和 Amazon ECS 入門。
  - 對於包括 Amazon EKS 在內的 Kubernetes，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 App Mesh 控制器或教學課程：設定與 Kubernetes 的應用程 App Mesh 整合。
  - 對於 Amazon EC2，請確保您已設置 Amazon EC2 實例以代理應 App Mesh 流量。如需詳細資訊，請參閱更新服務。
- 請確定運算服務上執行的 Envoy 容器已成功連線至 App Mesh Envoy 管理服務。您可以通過檢查該字`control_plane.connected_state`段的特使統計數據來確認這一點。如需詳細資訊`control_plane.connected_state`，請參閱我們的疑難排解最佳實務中的監控 Envoy Proxy 連線。

  如果特使最初能夠建立連線，但後來中斷連線且從未重新連線，請參閱 Envoy 與 App Mesh Envoy 管理服務中斷連線，並顯示錯誤文字，以解決其中斷連線的原因。

如果應用程式連線，但要求失敗並顯示`HTTP 503`回應碼，請嘗試下列動作：

- 請確定您要連線的虛擬服務存在於網狀中。
- 請確定虛擬服務具有提供者 (虛擬路由器或虛擬節點)。

- 使用 Envoy 作為 HTTP 代理時，如果您看到的是通過 Envoy 統計信息進入的出口流量`cluster.cds_egress_*_mesh-allow-all`而不是正確的目的地，則很可能 Envoy 沒有通過正確路由請求。`filter_chains`這可能是因為使用不合格的虛擬服務名稱所致。建議您使用實際服務的服務探索名稱作為虛擬服務名稱，因為 Envoy Proxy 會透過其名稱與其他虛擬服務進行通訊。

  如需詳細資訊，請參閱虛擬服務。
- 檢查 Envoy 代理記錄檔是否有下列任何錯誤訊息：
  - `No healthy upstream`— Envoy Proxy 嘗試路由到的虛擬節點沒有任何已解析的端點，或沒有任何正常運作的端點。確定目標虛擬節點具有正確的服務探索和健全狀況檢查設定。

    如果在後端虛擬服務的部署或擴展期間，對服務的要求失敗，請遵循中的指導當虛擬服務具有虛擬節點提供者503時，某些請求會失敗，並顯示 HTTP 狀態碼。
  - `No cluster match for URL`— 當要求傳送至不符合虛擬路由器提供者下定義的任何路由所定義之準則的虛擬服務時，這很可能是造成的。確定應用程式的要求會傳送至支援的路由，方法是確定路徑和 HTTP 要求標頭是正確的。
  - `No matching filter chain found`— 當要求傳送至無效連接埠上的虛擬服務時，很可能會造成這種情況。請確定來自應用程式的要求使用虛擬路由器上指定的相同連接埠。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法連線至外部服務

徵狀

您的應用程式無法連線至網狀外部的服務，例如`amazon.com`。

解析度

根據預設，App Mesh 不允許從網狀內的應用程式傳出流量到網狀外的任何目的地。若要啟用與外部服務的通訊，有兩個選項：

- 將網格資源上的輸出篩選器設定為`ALLOW_ALL`。此設定將允許網狀內的任何應用程式與網格內部或外部的任何目標 IP 位址進行通訊。
- 使用虛擬服務、虛擬路由器、路由和虛擬節點在網狀中建立外部服務的模型。例如，若要建立外部服務的模型`example.com`，您可以建立以虛擬路由器命名`example.com`的虛擬服務，並將所有流量傳送至具有 DNS 服務探索主機名稱為的虛擬節點`example.com`。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法連接到 MySQL 或 SMTP 伺服器

徵狀

當允許使用虛擬節點定義傳輸至所有目的地 (Mesh `EgressFilter type =ALLOW_ALL`) (例如 SMTP 伺服器或 MySQL 資料庫) 時，來自應用程式的連線會失敗。作為一個例子，以下是嘗試連接到 MySQL 服務器的錯誤消息。

```
ERROR 2013 (HY000): Lost connection to MySQL server at 'reading initial communication
 packet', system error: 0
```

解析度

這是使用應用程式網狀影像版本 1.15.0 或更新版本可解決的已知問題。如需詳細資訊，請參閱無法使 App Mesh 連線到 MySQL GitHub 問題。發生此錯誤是因為應用程式網格設定的 Envoy 中的輸出接聽程式新增了 Envoy TLS Inspector 器接聽程式篩選器。如需詳細資訊，請參閱特使文件中的 TLS Inspector 查器。此篩選器會檢查從用戶端傳送的第一個封包，來評估連線是否使用 TLS。但是，對於 MySQL 和 SMTP，服務器在連接後發送第一個數據包。如需有關 MySQL 的詳細資訊，請參閱 MySQL 文件中的初始握手。由於伺服器傳送第一個封包，因此篩選器的檢查會失敗。

要根據您的 Envoy 版本解決此問題：

- 如果您的 App Mesh 映像特使版本為 1.15.0 或更新版本，請勿將 MySQL、SMTP、MSSQL 等外部服務建立為應用程式虛擬節點的後端模型。
- 如果您的 App Mesh 映像特使版本是 1.15.0 之前，請將端口添加**3306**到 MySQL 服務**APPMESH_EGRESS_IGNORED_PORTS**中的值列表以及用於 STMP 的端口。

> **Important**
>
> 雖然標準 SMTP 通訊埠是25、和 587465，但您應該只新增目前使用的連接埠，`APPMESH_EGRESS_IGNORED_PORTS`而不是全部新增這三個連接埠。

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。

如果您的問題仍未解決，您可以向我們提供有關使用現有GitHub 問題所遇到的詳細資訊，或聯絡 Sup AWSport 部門。

無法連接到在 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。

解析度

這是已知問題。如需詳細資訊，請參閱路由至上相同連接埠上的多個 TCP 目的地 GitHub。由於在 OSI 第 4 層提供給特使代理的信息中的限制，App Mesh 目前不允許模型為 TCP 的多個後端目標共享相同的端口。若要確定 TCP 流量可以針對所有後端目的地適當路由，請執行下列動作：

- 確定所有目的地都使用唯一的連接埠。如果您使用虛擬路由器提供者進行後端虛擬服務，您可以變更虛擬路由器連接埠，而無需變更其路由到的虛擬節點上的連接埠。這可讓應用程式在 Envoy Proxy 繼續使用虛擬節點中定義的連接埠時，開啟虛擬路由器連接埠上的連線。
- 如果模型為 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。無法連接到 MySQL 或 SMTP 伺服器

如果您的問題仍未解決，您可以向我們提供有關使用現有GitHub 問題所遇到的詳細資訊，或聯絡 Sup AWSport 部門。

連線成功，未列為虛擬節點之虛擬服務後端的服務

徵狀

您的應用程式能夠將流量連線並傳送至虛擬節點上未指定為虛擬服務後端的目的地。

解析度

如果要求成功傳送至尚未在 App Mesh API 中建模的目的地，則最有可能的原因是網格的輸出篩選器類型已設定為。ALLOW_ALL當輸出篩選器設定為時ALLOW_ALL，應用程式的輸出要求將不符合模型化目的地 (後端) 的輸出要求傳送至應用程式設定的目的地 IP 位址。

如果您想要禁止流量傳送至未在網格中建模的目的地，請考慮將輸出篩選器值設定為。DROP_ALL

> **Note**
>
> 設定網格輸出篩選器值會影響網格內的所有虛擬節點。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

當虛擬服務具有虛擬節點提供者503時，某些請求會失敗，並顯示 HTTP 狀態碼

徵狀

部分應用程式的要求無法傳送至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。使用虛擬路由器提供者進行虛擬服務時，要求不會失敗。

解析度

這是已知問題。如需詳細資訊，請參閱 < 虛擬服務的虛擬節點提供者上的重試原則 > GitHub。使用虛擬節點做為虛擬服務的提供者時，您無法指定要讓虛擬服務用戶端使用的預設重試原則。相較之下，虛擬路由器提供者允許指定重試原則，因為它們是子路由資源的屬性。

若要減少虛擬節點提供者的要求失敗，請改用虛擬路由器提供者，並在其路由上指定重試原則。如需減少應用程式要求失敗的其他方法，請參閱App Mesh 最佳實務。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法連線到 Amazon EFS 檔案系統

徵狀

將 Amazon ECS 任務設定為磁碟區時，將 Amazon EFS 檔案系統設定為磁碟區時，工作無法啟動，並出現以下錯誤。

```
ResourceInitializationError: failed to invoke EFS utils commands to set up EFS volumes:
 stderr: mount.nfs4: Connection refused : unsuccessful EFS utils command execution;
 code: 32
```

解析度

這是已知問題。之所以發生這個錯誤，是因為 Amazon EFS 的 NFS 連線是在任務中的任何容器啟動之前發生的。此流量會由 Proxy 組態路由至 Envoy，此時不會執行。由於啟動的順序，NFS 用戶端無法連線至 Amazon EFS 檔案系統，且工作無法啟動。若要解決此問題，請將連接埠新增2049至 Amazon ECS 任務定義之 Proxy 組態中的`EgressIgnoredPorts`設定值清單。如需詳細資訊，請參閱 Proxy 組態。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

連線成功服務，但傳入的要求不會出現在 Envoy 的存取記錄、追蹤或指標中

徵狀

即使您的應用程式可以連線並傳送要求至其他應用程式，您仍無法在存取記錄檔或追蹤 Envoy Proxy 的資訊中看到傳入要求。

解析度

這是已知問題。從更多信息，請參閱 Github 上的 iptables 規則設置問題。Envoy Proxy 只會攔截其對應虛擬節點所偵聽之連接埠的入埠流量。對任何其他端口的請求將繞過 Envoy 代理並直接訪問其後面的服務。為了讓 Envoy 代理攔截服務的入站流量，您需要將虛擬節點和服務設置為在同一個端口上監聽。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

在容器級別設置HTTP_PROXY/HTTPS_PROXY環境變量不能按預期工作。

徵狀

當代理伺服器在以下位置設定為環境變數時：

- 啟用 App Mesh 的任務定義中的應用程序容器，發送到 App Mesh 服務命名空間的請求將從 Envoy 側車獲得HTTP 500錯誤響應。
- 啟用 App Mesh 的任務定義中的 Envoy 容器，從 Envoy 側車發出的請求將不會通過HTTP/HTTPS代理服務器，並且環境變量將無法正常工作。

解析度

對於應用程序容器：

通過使任務中的流量通過 Envoy 代理進行 App Mesh 功能。 HTTP_PROXY/HTTPS_PROXY配置通過配置容器流量通過不同的外部代理覆蓋此行為。Envoy 仍會攔截流量，但不支援使用外部 Proxy 代理網狀流量。

如果您想要代理所有非網狀流量，請設定NO_PROXY為包含網狀的 CIDR/ 命名空間、localhost 和認證的端點，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
```

對於特使容器：

特使不支持通用代理。我們不建議設定這些變數。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

即使在設置路由的超時後，上游請求超時。

徵狀

您已定義下列項目的逾時：

- 路由，但您仍然收到上游請求超時錯誤。
- 虛擬節點接聽程式以及路由的逾時和重試逾時，但您仍會收到上游要求逾時錯誤。

解析度

若要成功完成超過 15 秒的高延遲要求，您必須同時在路由和虛擬節點接聽程式層級指定逾時。

如果您指定的路由逾時大於預設 15 秒，請確定也為所有參與虛擬節點的監聽器指定逾時。但是，如果您將逾時減少為低於預設值的值，則可選擇更新虛擬節點上的逾時。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。

如果您指定重試原則，則您為要求逾時指定的持續時間永遠大於或等於重試逾時，乘以您在重試原則中定義的重試次數上限。這可讓您成功完成所有重試的要求。如需詳細資訊，請參閱路由。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

特使用 HTTP 錯誤請求進行響應。

徵狀

對於透過 Network Load Balancer (NLB) 傳送的所有要求，特使會回應 HTTP 400 錯誤要求。當我們檢查特使日誌時，我們看到：

- 偵錯記錄檔：

```
dispatch error: http/1.1 protocol error: HPE_INVALID_METHOD
```

- 存取記錄：

```
"- - HTTP/1.1" 400 DPE 0 11 0 - "-" "-" "-" "-" "-"
```

解析度

解決方案是停用 NLB 目標群組屬性上的代理通訊協定第 2 版 (PPv2)。

到目前為止，PPV2 不受使用 App Mesh 控制平面運行的虛擬網關和虛擬節點特使的支持。如果您使用 Kubernetes 上的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 AWSLoad Balancer 控制器註釋。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

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，以及通往虛擬節點後端的路由逾時，您的要求仍會在 15 秒內逾時。

解析度

請確定後端清單下包含正確的虛擬服務。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

App Mesh 縮放疑

本主題詳細說明您在使用 App Mesh 縮放時可能會遇到的常見問題。

當虛擬節點/虛擬閘道擴充超過 50 個複本時，連線會失敗且容器健康狀態檢查失敗

徵狀

當您擴展虛擬節點/虛擬閘道超過 50 個的複本數量時，例如 Amazon ECS 任務、Kubernetes 網繭或 Amazon EC2 執行個體，特使容器運作狀態檢查是否有新的和目前執行中的 Envoys 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具有 HTTP 狀態碼的要求失敗。`503`

解析度

每個虛擬節點/虛擬閘道的使用者數量，App Mesh 的預設配額為 50。當執行中的特使人數超過此配額時，新增和目前執行中的特使將無法使用 gRPC 狀態碼連線至 App Mesh 的特使管理服務 ()。8 RESOURCE_EXHAUSTED這個配額可以提高。如需詳細資訊，請參閱 App Mesh。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

503當虛擬服務後端水平向外擴展或擴展時，請求失敗

徵狀

當後端虛擬服務水平擴充或放入時，來自下游應用程式的要求會失敗，並HTTP 503顯示狀態碼。

解析度

App Mesh 建議使用數種方法，以減輕水平擴展應用程式時的故障情況 如需如何防止這些失敗的詳細資訊，請參閱App Mesh 最佳實務。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

特使容器在負載增加下崩潰與段錯誤

徵狀

在高流量負載下，Envoy 代理由於分段錯誤（Linux 退出代碼139）而崩潰。特使流程記錄檔包含如下所示的陳述式。

```
Caught Segmentation fault, suspect faulting address 0x0"
```

解析度

Envoy 代理伺服器可能違反了作業系統的預設無檔案 ulimit，也就是處理序一次可以開啟的檔案數量的限制。此漏洞是由於流量導致更多連接，從而消耗了額外的操作系統套接字。若要解決此問題，請增加主機作業系統上的 ulimit 無檔案值。如果您使用的是 Amazon ECS，則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。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

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。

解析度

雖然目前未顯示新的限制，但客戶仍然可以行使這些限制。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

由於大量運行狀態檢查呼叫，應用程式崩潰。

徵狀

啟用虛擬節點的作用中健全狀況檢查後，健康狀態檢查呼叫的數目會上升。由於對應用程式進行的健康狀態檢查呼叫量大幅增加，應用程式崩潰。

解析度

啟用主動健康狀態檢查後，下游 (用戶端) 的每個 Envoy 端點都會將健康狀態要求傳送至上游叢集 (伺服器) 的每個端點，以便做出路由決策。因此，健康檢查要求的總數為 `number of client Envoys number of server Envoys * active health check frequency`。

若要解決這個問題，請修改健全狀況檢查探查的頻率，這樣會減少健全狀況檢查探查的總數量。除了主動的健康狀態檢查之外，App Mesh 還允許將離群值偵測設定為被動健康狀態檢查的方法。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。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

App Mesh 觀測性疑難

本主題詳細說明使用 App Mesh 可觀察性時可能會遇到的常見問題。

無法看到我的應用程序的AWS X-Ray跟踪

徵狀

您在 App Mesh 中的應用程式未在 X-Ray 主控台或 API 中顯示 X-Ray 追蹤資訊。

解析度

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。請執行以下步驟，確認 X-Ray 已正確設定：

- 請確定 App Mesh 虛擬節點接聽程式通訊協定未設定為`TCP`。

- 確保隨應用程序一起部署的 X-Ray 容器公開 UDP 端口2000並以用戶身份運行。1337如需詳細資訊，請參閱上的 Amazon ECS X-Ray 範例。 GitHub
- 確定 Envoy 容器已啟用追蹤。如果您正在使用 App Mesh Envoy 映像檔，則可以將環境變數設定為的值，將ENABLE_ENVOY_XRAY_TRACING環境變數設定為來2000啟用 X-Ray。1 XRAY_DAEMON_PORT
- 如果您已使用其中一個特定語言 SDK 在應用程式程式碼中檢測 X-Ray，請遵循您所使用語言的指南，確定已正確設定 X 射線。
- 如果所有先前的項目都設定正確，請檢閱 X-Ray 容器記錄檔是否有錯誤，並遵循疑難排解中的指引 AWS X-Ray。有關 App Mesh 中 X-Ray 集成的更詳細說明可以在將 X-Ray 與 App Mesh 集成中找到。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法在 Amazon CloudWatch 指標中查看我應用程式的特使指標

徵狀

您在 App Mesh 中的應用程式不會將 Envoy 代理產生的指標發送至 CloudWatch指標。

解析度

在 App Mesh 中使用 CloudWatch 指標時，必須正確設定數個元件，以啟用 Envoy Proxy、CloudWatch 代理程式附屬和 CloudWatch 指標服務之間的通訊。請執行下列步驟，確認 Envoy 代理的 CloudWatch 指標是否已正確設定：

- 請確定您正在使用應用程式網格的 CloudWatch 代理程式映像。如需詳細資訊，請參閱開啟的應用程CloudWatch式網格 GitHub代理
- 請確定您已依照平台特定的使用指示，適當地設定 App Mesh 的 CloudWatch 代理程式。如需詳細資訊，請參閱開啟的應用程 CloudWatch式網格 GitHub代理
- 如果所有先前項目都設定正確，請檢閱 CloudWatch 代理程式容器記錄檔中是否有錯誤，並遵循疑難排解 CloudWatch 代理程式中提供的指引。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

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。

解析度

由於 App Mesh Envoy 目前不支援動態 X-Ray 取樣配置，因此可以使用以下解決方法。

如果您的 Envoy 版本是1.19.1或更新版本，則可以選擇以下選項。

- 若只要設定取樣率，請使用 Envoy 容器上的XRAY_SAMPLING_RATE環境變數。該值應指定為介於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開發人員指南中的抽樣規則。

如果您的 Envoy 版本較早1.19.1，請執行以下操作。

- 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 特使配置變量。指定自訂追蹤組態並定義本機取樣規則。有關更多信息，請參閱特使 X-Ray 配置。
- 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
```

- 如需有關內samplingRuleManifest容中取樣規則資訊清單設定的詳細資訊，請參閱針對 Go 設定 X-Ray SDK。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

App Mesh 安全性

本主題詳細說明使用 App Mesh 安全性時可能會遇到的常見問題。

無法使用 TLS 用戶端原則連線到後端虛擬服務

徵狀

將 TLS 用戶端原則新增至虛擬節點中的虛擬服務後端時，該後端的連線會失敗。嘗試傳送流量至後端服務時，要求會失敗，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`。

解析度

為了判斷問題的根本原因，我們建議您使用 Envoy Proxy 處理程序記錄檔來協助您診斷問題。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：

- 通過排除中無法連線至虛擬服務後端提到的錯誤，確保與後端的連接成功。
- 在 Envoy 處理程序記錄中，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：

- 憑證未由 TLS 用戶端原則信任服務包中定義的其中一個憑證授權單位簽署。
- 憑證已不再有效 (已過期)。
- 主體別名 (SAN) 與要求的 DNS 主機名稱不符。
- 請確定後端服務提供的憑證是有效的、由 TLS 用戶端原則信任服務包中的其中一個憑證授權單位簽署，並且符合中定義的準則Transport Layer Security (TLS)。
- 如果您收到的錯誤與下面的錯誤類似，則意味著請求繞過 Envoy 代理並直接到達應用程序。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 Proxy 組態中，請確定`AppPorts`包含應用程式偵聽的正確值。

```
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
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。如果您認為自己發現了安全漏洞或對 App Mesh 的安全性有疑問，請參閱AWS漏洞報告指南。

當應用程式原始 TLS 時，無法連線至後端虛擬服務

徵狀

從應用程式 (而非使用 Envoy Proxy) 來源 TLS 工作階段時，連線至後端虛擬服務會失敗。

解析度

這是已知問題。如需詳細資訊，請參閱功能要求：下游應用程式與上游 Proxy GitHub 問題之間的 TLS 交涉。在 App Mesh 中，目前支援特使代理伺服器的 TLS 產生，但不支援應用程式。若要使用特使的 TLS 起始支援，請在應用程式中停用 TLS 起始。這可讓 Envoy 讀取輸出要求標頭，並透過 TLS 工作階段將要求轉寄至適當的目的地。如需詳細資訊，請參閱 Transport Layer Security (TLS)。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。如果您認為自己發現了安全漏洞或對 App Mesh 的安全性有疑問，請參閱AWS漏洞報告指南。

無法聲明特使代理之間的連接正在使用 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 用戶端原則上啟用 TLS 起始，但是您無法宣告 Envoy Proxy 之間的連線正在透過 TLS 交涉的工作階段發生。

解析度

此解決方案中定義的步驟會利用 Envoy 管理介面和特使統計資料。如需設定這些項目的說明，請參閱啟用特使代理管理介面和為量度卸載啟用特使 DogStats D 整合。為了簡化起見，下列統計資料範例使用管理介面。

- 對於執行 TLS 終止的特使代理：
  - 使用下列命令確定 TLS 憑證已在 Envoy 組態中啟動載入。

    ```
    curl http://my-app.default.svc.cluster.local:9901/certs
    ```

    在傳回的輸出中，您應該會在下看到 TLS 終止中使`certificates[].cert_chain`用之憑證的至少一個項目。

- 請確定 Proxy 監聽器的成功輸入連線數目與 SSL 交握數目加上重複使用的 SSL 工作階段數目完全相同，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)
```

- 對於執行 TLS 產生的特使代理：
  - 使用下列命令確定 TLS 信任存放區已在 Envoy 組態中啟動載入。

```
curl http://my-app.default.svc.cluster.local:9901/certs
```

    在 TLS 產生期間，您應該至少會在下`certificates[].ca_certs`看到一個用於驗證後端憑證的憑證項目。

  - 請確定成功輸出連線至後端叢集的數目與 SSL 交握數目加上重複使用的 SSL 工作階段數目完全相同，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)
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。如果您認為自己發現了安全漏洞或對 App Mesh 的安全性有疑問，請參閱AWS漏洞報告指南。

使用 Elastic Load Balancing 疑難排解 TLS

徵狀

嘗試將應用程式負載平衡器或 Network Load Balancer 設定為加密虛擬節點的流量時，連線和負載平衡器健康狀態檢查可能會失敗。

解析度

為了確定問題的根本原因，您需要檢查以下內容：

- 對於執行 TLS 終止的特使代理，您需要排除任何錯誤的配置。請遵循上述中提供的步驟無法使用 TLS 用戶端原則連線到後端虛擬服務。
- 對於負載平衡器，您需要查看 `TargetGroup`：
  - 請確定`TargetGroup`連接埠符合虛擬節點定義的監聽程式連接埠。
  - 對於透過 HTTP 來源 TLS 連線至您的服務的應用程式負載平衡器，請確`TargetGroup`定通訊協定已設定為。`HTTPS`如果正在使用健康狀態檢查，請確定`HealthCheckProtocol`已將其設定為`HTTPS`。
  - 對於透過 TCP 來源 TLS 連線至服務的網路負載平衡器，請確`TargetGroup`定通訊協定已設定為。`TLS`如果正在使用健康狀態檢查，請確定`HealthCheckProtocol`已將其設定為`TCP`。

> **Note**
>
> 任何`TargetGroup`需要更改`TargetGroup`名稱的更新。

如果設定正確，您的負載平衡器應該使用提供給 Envoy Proxy 的憑證，為您的服務提供安全連線。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。如果您認為自己發現了安全漏洞或對 App Mesh 的安全性有疑問，請參閱AWS漏洞報告指南。

App Mesh 清除疑難排解

本主題詳細說明您在搭配 Kubernetes 使用應用程式網格時可能會遇到的常見問題。

在 App Mesh 中找不到在 Kubernetes 中建立的應用程式網格資源

徵狀

您已使用 Kubernetes 自訂資源定義 (CRD) 建立 App Mesh 資源，但是當您使用或 API 時，您建立的資源不會顯示在應用程式網格中。AWS Management Console

解析度

可能的原因是 App Mesh 的 Kubernetes 控制器中的錯誤。如需詳細資訊，請參閱上的疑難排解 GitHub。檢查控制器記錄檔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)
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

注入特使側車後，豆莢正在失敗準備和活力檢查

徵狀

您應用程式的 Pod 先前已成功執行，但是在將 Envoy 側車插入 Pod 之後，準備程度和活動性檢查開始失敗。

解析度

確保插入到網繭中的特使容器已透過 App Mesh 的特使管理服務啟動載入。您可以參考中的錯誤代碼來驗證任何錯誤特使與 App Mesh 特使管理服務斷開連接，並顯示錯誤文本。您可以使用下列命令來檢查相關網繭的 Envoy 記錄。

```
kubectl logs -n appmesh-system -f \
    $(kubectl get pods -n appmesh-system -o name | grep controller) \
    | grep "gRPC config stream closed"
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

Pod 未註冊或取消註冊為AWS Cloud Map執行個體

徵狀

您的 Kubernetes 網繭並未在其生命週期中註冊或取消註冊。AWS Cloud Map網繭可能會成功啟動並準備好為流量提供服務，但不會接收任何流量。當網繭終止時，用戶端可能仍會保留其 IP 位址並嘗試傳送流量至該網繭，導致失敗。

解析度

這是已知問題。如需詳細資訊，請參閱在 Kubernetes 中發生問題的網繭無法 auto 註冊/取消註冊。AWS Cloud Map GitHub 由於網繭、App Mesh 虛擬節點和AWS Cloud Map資源之間的關係，Kubernetes 的 App Mesh 控制器可能會變得不同步處理並失去資源。例如，如果在終止其關聯的網繭之前從 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：

- 請確定您正在執行適用於 Kubernetes 的 App Mesh 控制器的最新版本。
- 請確定虛擬節點定義中的AWS Cloud Map`namespaceName`和`serviceName`是正確的。
- 在刪除虛擬節點定義之前，請確定已刪除任何關聯的網繭。如果您需要協助識別哪些網繭與虛擬節點相關聯，請參閱無法判斷應用程式網格資源的網繭在何處執行。
- 如果問題仍然存在，請執行下列命令來檢查控制器記錄檔是否有助於揭示潛在問題的錯誤。

```
kubectl logs -n appmesh-system \
    $(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

- 請考慮使用下列命令重新啟動控制器 Pod。這可能會修正同步處理問題。

```
kubectl delete -n appmesh-system \
    $(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法判斷應用程式網格資源的網繭在何處執行

徵狀

當您在 Kubernetes 叢集上執行 App Mesh 時，操作員無法判斷指定應用程式網格資源的工作負載或網繭在何處執行。

解析度

Kubernetes 網繭資源會以其相關聯的網格和虛擬節點加上註解。您可以使用下列命令查詢針對指定虛擬節點名稱執行的網繭。

```
kubectl get pods --all-namespaces -o json | \
    jq '.items[] | { metadata } | select(.metadata.annotations."appmesh.k8s.aws/
virtualNode" == "virtual-node-name")'
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

無法判斷網繭執行的應用程式網格資源為何

徵狀

在 Kubernetes 叢集上執行 App Mesh 時，操作員無法判斷指定網繭所執行的應用程式網狀資源為何。

解析度

Kubernetes 網繭資源會以其相關聯的網格和虛擬節點加上註解。您可以使用下列命令直接查詢網繭，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" }'
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

在停用 IMDSv1 的情況下，用戶端使用者無法與 App Mesh 特使管理服務進行通訊

徵狀

停IMDSv1用時，用戶端使者無法與 App Mesh 控制平面 (特使管理服務) 通訊。IMDSv2之v1.24.0.0-prod前在 App Mesh 特使版本上不提供支持。

解析度

要解決此問題，您可以執行以下三件事之一。

- 升級到具有IMDSv2支持的 App Mesh 特使版本v1.24.0.0-prod或更高版本。

- 在執行 Envoy IMDSv1 的執行個體上重新啟用。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，建議您針對服務帳戶 (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。

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

啟用應用程序網格並注入特使時，IRSA 不適用於應用程序容器

徵狀

在 Amazon EKS 的應用程式網狀控制器的協助下在 Amazon EKS 叢集上啟用應用程式網狀結構時，會將特使和proxyinit容器插入到應用程式網繭中。應用程式無法假設，IRSA而是假設node role. 當我們描述 Pod 詳細資料時，我們會看到AWS_WEB_IDENTITY_TOKEN_FILE或AWS_ROLE_ARN環境變數未包含在應用程式容器中。

解析度

如果已定義AWS_WEB_IDENTITY_TOKEN_FILE或AWS_ROLE_ARN環境變數，則 webhook 會略過網繭。不要提供這些變量中的任何一個，webhook 將為您注入它們。

```
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
        }
```

如果您的問題仍然無法解決，請考慮開啟GitHub 問題或連絡 Sup AWSport 部門。

App Mesh

應用程式網狀預覽通道是us-west-2區域中提供的 App Mesh 服務的獨特變體。預覽頻道會公開即將推出的功能，供您在開發過程中嘗試使用。當您在預覽頻道中使用功能時，您可以透過提供意見反應，GitHub 以塑造特徵的行為方式。一旦功能在預覽頻道中具有完整功能，並且完成了所有必要的整合和檢查，它將畢業到生產 App Mesh 服務。

AWS App Mesh預覽頻道是 Beta 版服務，所有功能均為預覽版，因為這些條款已在AWS服務條款中定義。您對預覽頻道的參與受到您的協議AWS和AWS服務條款的約束，尤其是通用和 Beta 版服務參與條款，並且是機密的。

以下是關於預覽頻道的常見問題。

什麼是預覽頻道？

Preview Channel 是一種公共服務端點，可讓您在新服務功能正式推出之前試用並提供意見反應。預覽通道的服務端點與標準生產端點不同。您可以使用預覽通道的服務模型檔案，以及AWS CLI.AWS CLI預覽通道可讓您在不影響目前的生產基礎架構的情況下嘗試新功能。我們鼓勵您向 App Mesh 團隊提供意見反應，以協助確保 App Mesh 符合客戶最重要的需求。您在預覽頻道中對功能的意見反應可協助塑造 App Mesh 的功能，讓我們能夠提供最佳服務。

預覽頻道與生產 App Mesh 有何不同？

下表列出了與預覽通道不同之 App Mesh 格服務的各個層面。

縱橫	App Mesh	App Mesh
Frontend endpoint	appmesh.us-west-2.amazonaws.com	appmesh-preview.us-west-2.amazonaws.com
Envoy management service endpoint	appmesh-envoy-management.us-west-2.amazonaws.com	appmesh-preview-envoy-management.us-west-2.amazonaws.com
CLI	AWS App Mesh list-meshes	AWS App Mesh-preview list-meshes (only available after adding the Preview Channel service model)

Signing name	appmesh	appmesh-preview
Service principal	appmesh.amazonaws.com	appmesh-preview.amazonaws.com

> **Note**
>
> 雖然 App Mesh 生產服務表格中的範例會列出us-west-2區域，但大部分的區域都可以使用生產服務。如需 App Mesh 生產服務可在其中使用的所有區域清單，請參閱AWS App Mesh端點和配額。不過，應用程式網狀預覽通道服務僅適用於該us-west-2地區。

如何使用預覽頻道中的功能？

1. AWS CLI使用下列命令將包含預覽頻道功能的預覽頻道服務模型新增至。

```
aws configure add-model \
    --service-name appmesh-preview \
    --service-model https://raw.githubusercontent.com/aws/aws-app-mesh-roadmap/
main/appmesh-preview/service-model.json
```

2. 根據 JSON 範例和功能使AWS App Mesh用者指南中提供的指示，建立包含該功能的 JSON 檔案。
3. 使用適當的AWS CLI命令和命令輸入文件實現該功能。例如，下列指令會使用 *route.json* 檔案建立具有預覽頻道功能的路由。

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

4. 將該容器新增到 Amazon ECS 任務定義、Kubernetes 網繭規格或 Amazon EC2 執行個體時，請新增APPMESH_PREVIEW = 1為特使容器的組態變數。此變數可讓 Envoy 容器與「預覽通道」端點進行通訊。如需新增組態變數的詳細資訊，請參閱更新 Amazon ECS 中的服務、在 Kubernetes 中更新服務和更新 Amazon EC2 上的服務。

如何提供意見回饋？

您可以直接針對 App Mesh 路線圖 GitHub存放庫問題提供意見反應，這些問題與功能有關的文件所連結。

我需要在預覽頻道中對某項功能提供意見反應多久？

回饋期間會依引入功能的大小和複雜度而變動。我們打算在將功能發布到預覽端點和將功能發布到生產之間提供 14 天的評論期。App Mesh 團隊可能會延長特定功能的意見回饋期限。

預覽頻道提供什麼等級的支援？

雖然我們鼓勵您直接就 App Mesh GitHub 路線圖問題提供意見反應和錯誤報告，但我們瞭解您可能需要分享敏感資料，或者您可能會發現您認為可以公開揭露的問題不安全。對於這些問題，您可以直接通過電子郵件向 App Mesh 團隊聯繫AWS Support或提供反饋。

我的資料在預覽通道端點上是否安全？

是。預覽通道的安全性等級與標準生產端點相同。

我的配置可以使用多久？

您可以在預覽通道中使用網格 30 天。建立網面三十天後，您只能列出、讀取或刪除網面。如果您在三十天後嘗試建立或更新資源，則會收到`BadRequest`例外狀況，說明網格已封存。

我可以使用哪些工具來處理預覽頻道？

您可以使AWS CLI用「預覽通道」服務模型檔案和指令輸入檔案。若要取得關於如何使用圖徵的詳細資訊，請參閱如何使用預覽頻道中的功能？。您無法使用AWS CLI指令選項AWS Management Console、SDK 或使AWS CloudFormation用「預覽頻道」功能。但是，一旦將特徵發佈到生產服務，您就可以使用所有工具。

預覽通道 API 是否具有向前相容性？

沒有 API 可能會根據意見反應而變更。

預覽頻道功能是否完成？

沒有 新的 API 物件可能未完全整合至AWS Management ConsoleAWS CloudFormation、或AWS CloudTrail。隨著功能在預覽頻道中固化，而且幾乎正式推出，整合最終將可用。

是否提供預覽頻道功能的說明文件？

是。預覽頻道功能的文件包含在生產文件中。例如，如果將路由資源的特徵釋放到「預覽通道」中，有關如何使用這些特徵的資訊將會在現有路由資源文件中提供。「預覽頻道」功能會標示為僅在「預覽頻道」中可用。

我如何知道預覽頻道何時有新功能可用？

在「預覽通道」中引入新功能時，會在「App Mesh 文件記錄」中新增一個項目。您可以定期檢閱頁面，或訂閱 App Mesh 文件歷史記錄 RSS 摘要。此外，您也可以檢閱AWS App Mesh藍圖 GitHub 存放庫的問題。預覽通道服務模型 JSON 檔案的下載連結會新增至發佈至預覽頻道時的問題。如需關於如何使用模型和特徵的詳細資訊，請參閱如何使用預覽頻道中的功能？。

我如何知道某個功能何時畢業於製作服務？

App Mesh 文件中指出該功能僅在「預覽通道」中可用的文字已移除，而且會在 App Mesh 文件記錄中新增項目。您可以定期檢閱頁面，或訂閱 App Mesh 文件歷史記錄 RSS 摘要。

App Mesh

AWS App Mesh已與 Service Quotas 整合，這是可讓您集中檢視和管理配額的AWS服務。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。

使用來檢視 App MeshAWS 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 Service quotas (服務配額) 清單中，您可以看到服務配額名稱、套用的值 (如果有的話)、AWS 預設配額，以及配額值是否可調整。
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。

使用來檢視 App MeshAWS 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
```

若要透過 AWS CLI 進一步使用服務配額，請參閱《Service Quotas AWS CLI 命令參考》。

App Mesh 的文件記錄

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
CloudTrail 集成文檔更新	說明應用程式網格整合 CloudTrail 以記錄 API 活動的說明文件已更新。	2024年3月28日
更新的政策	已更`AWSAppMeshServiceRolePolicy` 新`AWSServiceRoleForAppMesh` 並允許存取 AWS Cloud Map `DiscoverInstancesRevision` API。	2023 年 10 月 12 日
App Mesh 的 VPC 端點原則支援	App Mesh 現在支援 VPC 端點原則。	2023 年 5 月 11 日
App Mesh 的多個偵聽	App Mesh 現在支援多個偵聽程	2022 年 8 月 18 日
IPv6 的應用程式網格	App Mesh 現在支援 IPv6。	2022 年 5 月 18 日
CloudTrail App Mesh 特使管理服務的日誌記錄支持	App Mesh 現在支援 App Mesh 特使管理服務的 CloudTrail 記錄支援。	2022 年 3 月 18 日
特使的 App Mesh 代理	App Mesh 現在支持特使代理。	2022 年 2 月 25 日
App Mesh 的多個偵聽	(僅限應用程式網狀預覽通道) 您可以為 App Mesh 實作多個接聽程式。	2021 年 11 月 23 日
ARM64 對應用程式網格的支援	App Mesh 現在支援 ARM64。	2021 年 11 月 19 日

App Mesh 的度量延伸	您可以為 App Mesh 實作指標延伸模組。	2021 年 10 月 29 日
實作傳入流量增強	您可以針對主機名稱和路徑實作主機名稱和標頭比對和重寫。	2021 年 6 月 14 日
實作相互 TLS 驗證	您可以實作相互 TLS 驗證。	2021 年 2 月 4 日
地區在遠南 -1 啟動	App Mesh 現在可在遠南 -1 區域使用。	2021 年 1 月 22 日
實作相互 TLS 驗證	(僅限 App Mesh 預覽通道) 您可以實作相互 TLS 驗證。	2020 年 11 月 23 日
實作虛擬閘道接聽程式的連線集區	您可以為虛擬閘道接聽程式實作連線集區。	2020 年 11 月 5 日
實作虛擬節點接聽程式的連線集區和異常值偵測	您可以為虛擬節點接聽程式實作連線集區和異常值偵測。	2020 年 11 月 5 日
在歐洲南部 -1 區域啟動	App Mesh 現在可在歐洲南部 1 區域使用。	2020 年 10 月 21 日
實作虛擬閘道接聽程式的連線集區	(僅限 App Mesh 預覽通道) 您可以為虛擬閘道接聽程式實作連線集區。	2020 年 9 月 28 日
實作虛擬節點接聽程式的連線集區和異常值偵測	(僅限 App Mesh 預覽通道) 您可以為虛擬節點接聽程式實作連線集區和異常值偵測。	2020 年 9 月 28 日
為網格輸入建立虛擬閘道和閘道路由	啟用網格外的資源，以便與網格內的資源進行通訊。	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並進行路由	您可以將逾時值新增至虛擬節點接聽程式並進行路由。	2020 年 6 月 18 日
將逾時值新增至虛擬節點接聽程式	(僅限 App Mesh 預覽通道) 您可以將逾時值新增至虛擬節點接聽程式。	2020 年 5 月 29 日
為網格輸入建立虛擬閘道	(僅適用於 App Mesh 預覽通道) 啟用網狀外部的資源與網狀內部的資源通訊。	2020 年 4 月 8 日
TLS 加密	(僅限 App Mesh 預覽通道) 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
與另一個帳戶共用網格	(僅適用於 App Mesh 預覽通道) 您可以與其他帳戶共用網格。任何帳號建立的資源都可以與網格中的其他資源進行通訊。	2020 年 1 月 17 日
向路由添加超時值	(僅限 App Mesh 預覽通道) 您可以為路由新增逾時值。	2020 年 1 月 17 日
在 AWS 前哨上創建 App Mesh 代理	您可以在 AWS 前哨上創建 App Mesh 特使代理。	2019 年 12 月 3 日

HTTP/2 和 gRPC 支援路由器、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。您可以將重試邏輯添加到路由。	2019 年 9 月 10 日
TLS 加密	(僅限 App Mesh 預覽通道) 使用 TLS 加密虛擬節點之間的通訊。	2019 年 9 月 6 日
以 HTTP 標頭為基礎的路由	根據要求中 HTTP 標頭的存在狀態和值路由流量。	2019 年 8 月 15 日
應用 App Mesh 預覽通道的可用性	應用程式網狀預覽通道是 App Mesh 服務的不同變體。預覽頻道會公開即將推出的功能，供您在開發過程中嘗試使用。當您在預覽頻道中使用功能時，您可以透過提供意見反應，GitHub 以塑造特徵的行為方式。	2019 年 7 月 19 日
App Mesh 介面 VPC 端點 ()AWS PrivateLink	透過將 App Mesh 格設定為使用介面 VPC 端點，改善 VPC 的安全性狀態。介面端點採用這項技術 AWS PrivateLink，可讓您使用私有 IP 位址私有存取 App Mesh API。 PrivateLink 將 VPC 和應用程式網格之間的所有網路流量限制在 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 作為虛擬節點服務探索方法。若要用 AWS Cloud Map 於服務探索，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，建立應用程式網格資源，並自動將應用程式網格附加容器映像新增至您的 Kubernetes 部署。	2019 年 6 月 11 日
App Mesh 一般可用性	應用程式網格服務現已正式提供生產使用。	2019 年 3 月 27 日
App Mesh API 更新	應用程式網格 API 已更新，以改善可用性。有關更多信息，請參閱 [功能] 將監聽器添加到虛擬路由器和 [BUG] 路由到具有不匹配端口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。