aws

使用者指南

Amazon EventBridge

Amazon EventBridge: 使用者指南

Table of Contents

什麼是 Amazon EventBridge？

EventBridge 是一種無伺服器服務，該服務使用事件將應用程式元件連接在一起，讓您更輕鬆地建置可擴展的事件驅動型應用程式。事件驅動型架構是一種建置鬆耦合軟體系統的方式，透過發出和回應事件來協作。事件驅動架構可協助您提升靈活性，並建置可靠且可擴展的應用程式。

使用 EventBridge，將來自本土應用程式、AWS 服務和第三方軟體等來源的事件路由傳送至整個組織的消費者應用程式。EventBridge 提供簡單一致的方式來擷取、篩選、轉換和傳遞事件，讓您能夠快速建置應用程式。

以下影片為 Amazon EventBridge 功能簡介：

EventBridge 包含兩種事件處理方式：事件匯流排和管道。

- 事件匯流排是接收事件並將其傳遞至零個或多個目標的路由器。事件匯流排非常適合將事件從許多來源路由傳送至多個目標，可選擇在傳遞至目標之前轉換事件。

  以下影片給出了事件匯流排的高階概觀：

- 管道 EventBridge 管道適用於點對點整合；每個管道都會接收來自單一來源的事件，以進行處理，並將其傳遞至單一目標。管道還包括對進階轉換的支援，以及在傳遞至目標之前豐富事件。

管道和事件匯流排通常搭配使用。常見的使用案例是建立以事件匯流排為目標的管道；管道會將事件傳送至事件匯流排，然後將這些事件傳送至多個目標。例如，您可以建立一個管道，其中包含來源的 DynamoDB 串流，並建立事件匯流排作為目標。管道會從 DynamoDB 串流接收事件，並將其傳送至事件匯流排，然後根據您在事件匯流排上指定的規則，將事件傳送至多個目標。

EventBridge 是 Amazon CloudWatch Events 的新版本。

EventBridge 之前被稱為 Amazon CloudWatch Events。預設事件匯流排和您在 CloudWatch Events 中建立的規則也會顯示在 EventBridge 主控台中。EventBridge 使用相同的 CloudWatch Events API，因此使用 CloudWatch Events API 的程式碼保持不變。

EventBridge 以 CloudWatch Events 的功能為基礎，並提供合作夥伴事件、結構描述登錄檔和 EventBridge 管道等功能。新增至 EventBridge 的新功能不會新增至 CloudWatch Events。如需更多詳細資訊，請參閱 ???。

您在 CloudWatch Events 中使用的所有功能也會出現 EventBridge 中，包括：

- ???
- ???
- ???
- ???

建置和擴充事件功能的 EventBridge 功能包括：

- ???
- ???
- ???
- ???

Amazon EventBridge 設定和先決條件

若要使用 Amazon EventBridge，您需要 AWS 帳戶。您的帳戶可讓您使用 Amazon EC2 等服務來產生事件，然後在 EventBridge 主控台查看這些事件。您也可以安裝並設定 AWS Command Line Interface (AWS CLI) 以使用命令列介面來查看事件。

主題

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立管理使用者

在您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者、啟用 AWS IAM Identity Center，以及建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱AWS IAM Identity Center使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理存取權授予管理使用者。

   有關如何使用 IAM Identity Center 目錄 作為身分來源的教學課程，請參閱《AWS IAM Identity Center 使用者指南》中的以預設 IAM Identity Center 目錄 設定使用者存取權。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

登入 Amazon EventBridge 主控台

登入 Amazon EventBridge 主控台

- 登入 AWS Management Console，並在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。

帳戶憑證

雖然您可以使用根使用者憑證來存取 EventBridge，但我們建議您使用 AWS Identity and Access Management (IAM) 帳戶。如果您使用 IAM 帳戶來存取 EventBridge，您必須擁有以下許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "events:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:events:*:*:*"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "events.amazonaws.com"
        }
      }
    }
  ]
}
```

如需詳細資訊，請參閱 身分驗證。

設定 AWS Command Line Interface

您可使用 AWS CLI 來執行 EventBridge 操作。

如需如何安裝和設定 AWS CLI 的資訊，請參閱《AWS Command Line Interface 使用者指南》中的使用 AWS Command Line Interface 開始設定。

區域端點

您必須啟用預設的區域端點，以使用 EventBridge。如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS 區域內啟用和停用 AWS STS。

開始使用 Amazon EventBridge

的基礎 EventBridge 是建立將事件路由到目標的規則。在本節中，您可以建立基本規則。如需特定案例和特定目標的教學課程，請參閱 Amazon EventBridge 教學課程。

在 Amazon 中創建規則 EventBridge

若要為事件建立規則，您可以指定在 EventBridge 接收符合規則中事件模式的事件時要採取的動作。當事件相符時， EventBridge 會將事件傳送至指定的目標，並觸發規則中定義的動作。

當您 AWS 帳戶中的某個 AWS 服務發出事件時，它始終會進入您帳戶的預設事件匯流排。若要撰寫符合您帳戶中 AWS 服務之事件的規則，您必須將其與預設事件匯流排相關聯。

若要建立 AWS 服務的規則

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇規則。
3. 選擇建立規則。
4. 輸入規則的名稱和描述。

   在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
5. 針對事件匯流排，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 AWS 預設事件匯流排。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對規則類型，選擇具有事件模式的規則。
7. 選擇下一步。
8. 在事件來源欄位中，選擇 AWS 服務。
9. (選用) 針對範例事件，請選擇事件類型。
10. 針對事件模式，請執行下列其中一個動作：
    - 若要使用範本建立您的事件模式，請選擇事件模式表單，然後選擇事件來源和事件類型。如果您選擇「所有事件」作為事件類型，則此 AWS 服務發出的所有事件都會符合規則。

      若要自定範本，請選擇自訂模式 (JSON 編輯器)並進行變更。
    - 若要使用自訂事件模式，請選擇自訂模式(JSON 編輯器)並建立事件模式。
11. 選擇下一步。

12. 在目標類型欄位中，選擇 AWS 服務。
13. 針對 [選取目標]，選擇當 EventBridge 偵測到符合事件模式的事件時，要傳送資訊的 AWS 服務。
14. 顯示的欄位會根據您選擇的服務而有所不同。請視需要輸入此目標類型的特定資訊。
15. 對於許多目標類型， EventBridge 需要將事件傳送至目標的權限。在這些情況下， EventBridge 可以建立執行規則所需的 IAM 角色。執行以下任意一項：

    - 若要自動建立 IAM 角色，請選擇為此特定資源建立新角色。
    - 若要使用您早前建立的 IAM 角色，請選擇使用現有角色並從下拉式清單中選取現有角色。

16. (選用) 針對其他設定，請執行下列動作：

    a. 針對 Maximum age of event (事件的最長存留期)，輸入介於一分鐘 (00:01) 到 24 小時 (24:00) 之間的某個值。

    b. 針對重試嘗試，輸入介於 0 到 185 之間的某個數。

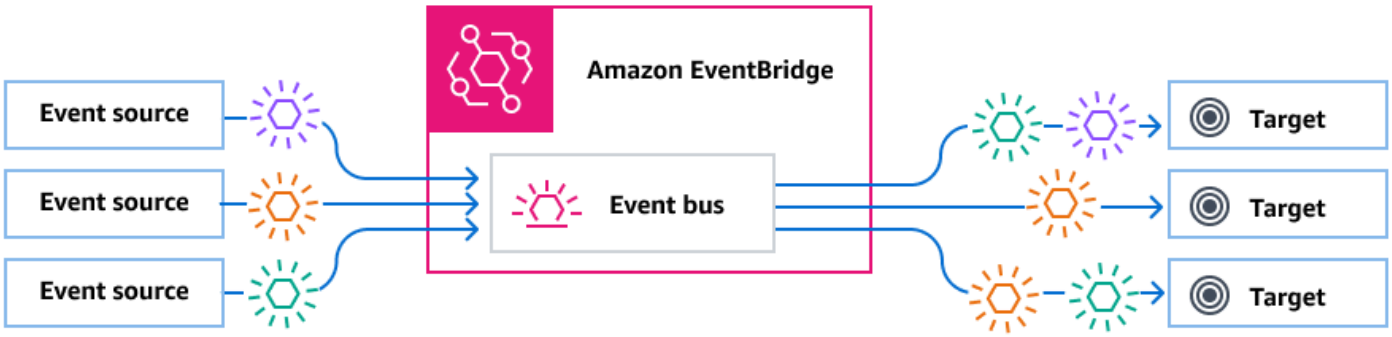
    c. 對於無效字母佇列，請選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。 EventBridge 如果符合此規則的事件未成功傳遞至目標，則會將符合此規則的事件傳送至無效字母佇列。執行以下任意一項：

       - 選擇無，即不使用無效字母佇列。
       - 選擇選取當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列，然後從下拉式清單中選取要使用的佇列。
       - 選擇選取其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列，然後輸入要使用的佇列 ARN。您必須將以資源為基礎的政策附加至佇列，以授與傳送訊息給該佇列的 EventBridge 權限。如需詳細資訊，請參閱 將許可授予無效字母佇列。

17. (選用) 選擇新增其他目標，為此規則新增另一個目標。
18. 選擇下一步。
19. (選用) 為規則輸入一或多個標籤。如需詳細資訊，請參閱 Amazon EventBridge 標籤。
20. 選擇下一步。
21. 檢閱規則的詳細資訊，然後選擇建立規則。

Amazon EventBridge 活動巴士

事件匯流排是接收事件並將事件傳遞至零個或多個目的地或目標的一種路由器。事件匯流排非常適合將事件從許多來源路由傳送至多個目標，可選擇在傳遞至目標之前轉換事件。



與事件匯流排建立關聯的規則會在事件到達時評估事件。每項規則都會檢查事件是否與規則模式相符。如果事件不匹配，則 EventBridge 發送事件

您可以將規則與特定事件匯流排相關聯，因此規則僅適用於該事件匯流排所接收的事件。

> **Note**
>
> 您也可以使用 EventBridge 管道處理事件。 EventBridge 管道用於 point-to-point 整合；每個管道都會接收來自單一來源的事件，以進行處理，並將其傳遞至單一目標。管道還包括對進階轉換的支援，以及在傳遞至目標之前豐富事件。如需詳細資訊，請參閱 ???。

主題

- 事件匯流排的運作方式
- Amazon EventBridge 事件總線概念
- 創建一個 Amazon EventBridge 事件總線
- 更新 Amazon EventBridge 事件總線
- 刪除 Amazon EventBridge 事件總線
- Amazon EventBridge 事件匯流排的許可
- 從 Amazon EventBridge 事件匯流排產生 AWS CloudFormation 範本

事件匯流排的運作方式

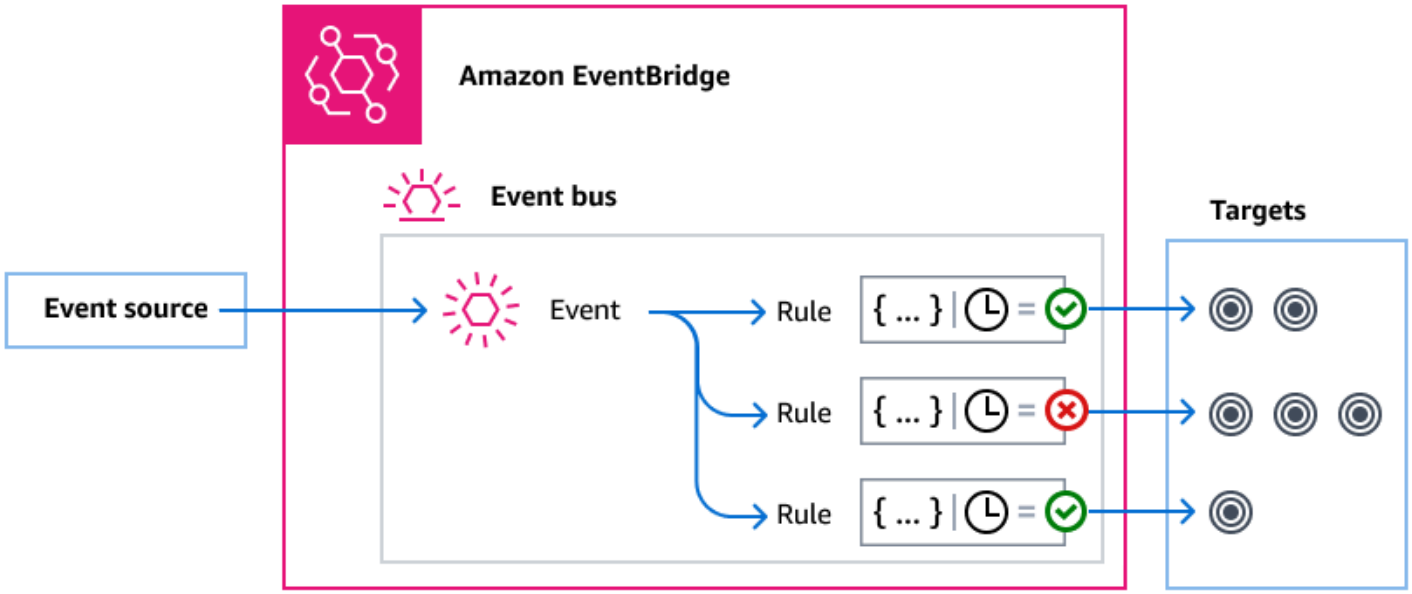
事件匯流排可讓您將事件從多個來源路由到多個目的地或目標。

在高階程序中，下面是它的運作方式：

1. 事件來源 (可以是 AWS 服務、您自己的自訂應用程式或 SaaS 提供者) 會將事件傳送至事件匯流排。
2. EventBridge 然後根據為該事件匯流排定義的每個規則評估事件。

   對於每個符合規則的事件， EventBridge 然後將事件傳送到為該規則指定的目標。或者，作為規則的一部分，您也可以指定在將事件傳送至目標之前 EventBridge 應如何轉換事件。

   一個事件可能符合多個規則，而每個規則最多可以指定五個目標。（事件可能不符合任何規則，在這種情況下不會 EventBridge 採取任何操作。）



假設使用 EventBridge 預設事件匯流排的範例，該匯流排會自動接收來自 AWS 服務的事件：

1. 您可以在 `EC2 Instance State-change Notification` 事件的預設事件匯流排上建立規則：
   - 您可以指定規則與 Amazon EC2 執行個體已變更 `state` 為 `running` 的事件相符。

     您可以透過指定 JSON 來定義事件必須符合的屬性和值，才能觸發規則。這就是所謂的事件模式。

```
{
  "source": ["aws.ec2"],
  "detail-type": ["EC2 Instance State-change Notification"],
    "detail": {
      "state": ["running"]
  }
}
```

- 您可以將規則的目標指定為指定的 Lambda 函數。

2. 每當 Amazon EC2 執行個體變更狀態時，Amazon EC2 (事件來源) 都會自動將該事件傳送到預設事件匯流排。
3. EventBridge 根據您建立的規則，評估傳送至預設事件匯流排的所有事件。

   如果事件符合您的規則 (也就是說，如果事件是將狀態變更為的 Amazon EC2 執行個體`running`)，則會將事件 EventBridge 傳送到指定的目標。在這種情況下，這是 Lambda 函數。

下列影片說明什麼是事件匯流排以及它們的作用：什麼是事件匯流排

以下視頻介紹了不同的事件匯流排以及何時使用它們：事件匯流排之間的差異

Amazon EventBridge 事件總線概念

以下是對在事件匯流排上建置的事件驅動架構的主要元件的進一步了解。

事件匯流排

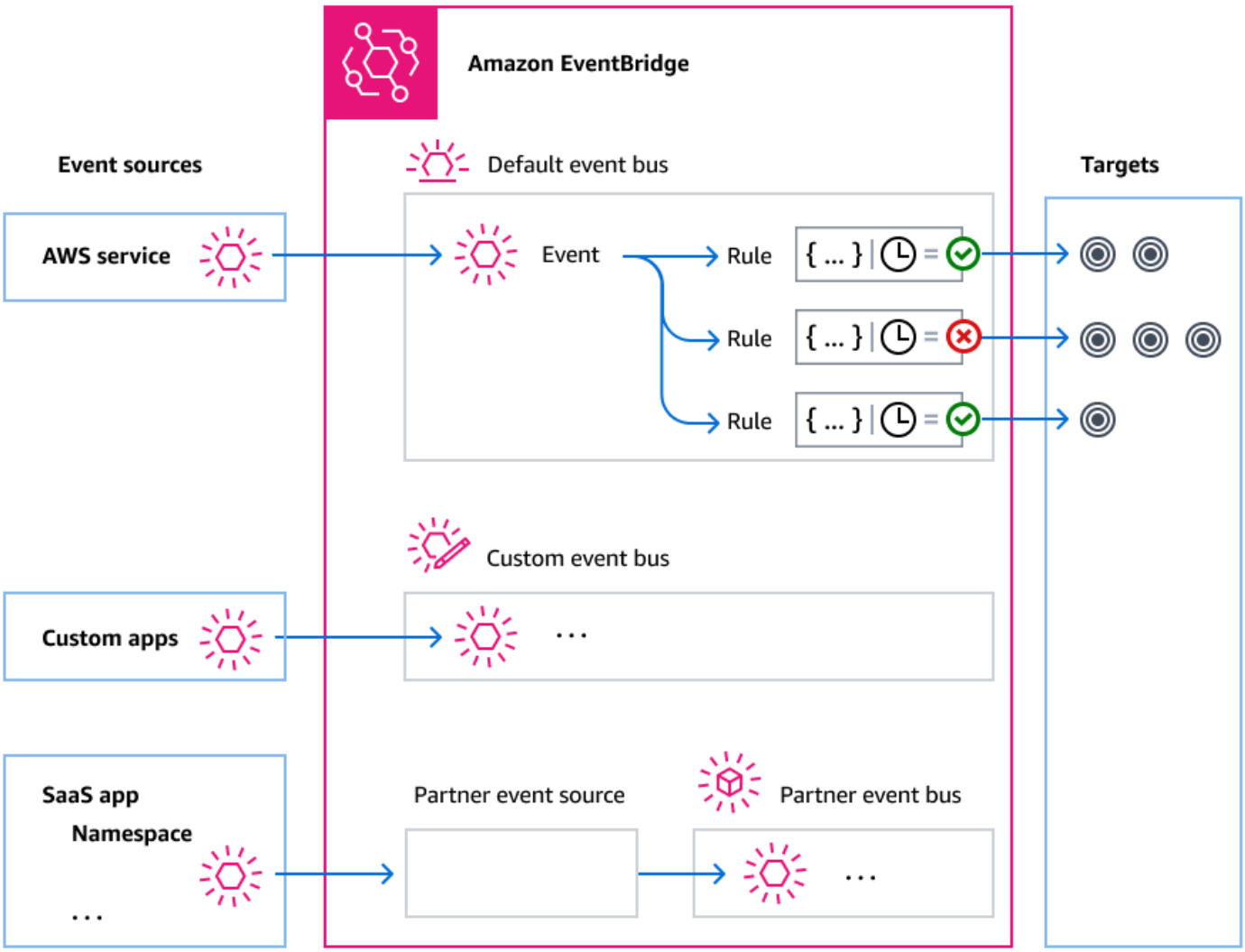
事件匯流排是接收事件並將事件傳遞至零個或多個目的地或目標的一種路由器。在您需要時使用事件匯流排將事件從許多來源路由傳送至多個目標，可選擇在傳遞至目標之前轉換事件。

您的帳戶包含預設事件匯流排，可自動接收來自 AWS 服務的事件。您也可以：

- 建立其他事件匯流排 (稱為自訂事件匯流排)，並指定它們接收的事件。
- 建立合作夥伴事件匯流排，以接收 SaaS 合作夥伴的事件。

事件匯流排的常見使用案例包括：

- 使用事件匯流排作為不同工作負載、服務或系統之間的代理程式。
- 在您的應用程式中使用多個事件匯流排來劃分事件流量。例如，建立一个匯流排以處理包含個人識別資訊 (PII) 的事件，而建立另一個匯流排用於處理沒有 PII 的事件。
- 透過將多個事件匯流排的事件傳送至集中式事件匯流排，以彙總事件。這個集中匯流排可以在與其他匯流排相同的帳戶中，也可以在不同的帳戶中。



事件

最簡單來說， EventBridge 事件是傳送至事件匯流排或管道的 JSON 物件。

在事件驅動架構 (EDA) 的前後關聯中，事件通常代表資源或環境中變更的指標。

如需詳細資訊，請參閱 ???。

事件來源

EventBridge 可以從事件來源接收事件，包括：

- AWS 服務
- 自訂應用程式
- 軟體即服務 (SaaS) 伙伴

規則

規則會接收傳入事件，並在適當的情況下將其傳送給目標以進行處理。您可以根據下列任一項指定每個規則調用其目標的方式：

- 事件模式，其中包含一個或多個篩選器來匹配事件。事件模式可以包含符合下列項目的篩選條件：
  - 事件中繼資料：事件相關資料，例如事件來源，或事件產生的帳戶或地區。
  - 事件資料：事件本身的屬性。這些屬性會根據事件而有所不同。
  - 事件內容：事件資料的實際屬性值。
- 定期調用目標的排程。

  您可以在中指定排程規則 EventBridge，也可以使用「EventBridge 排程器」來指定。

> **Note**
>
> EventBridge 提供 Amazon EventBridge Scheduler，這是一種無伺服器排程器，可讓您從單一中央受管服務建立、執行和管理任務。 EventBridge Scheduler 具有高度可自訂性，並提供比 EventBridge排程規則改善的可擴充性，並提供更廣泛的目標 API 作業和 AWS 服務。我們建議您使用 EventBridge 排程器依排程叫用目標。如需詳細資訊，請參閱 ???。

每個規則都是針對特定事件匯流排定義的，且僅適用於該事件匯流排上的事件。

單一規則最多可傳送五個目標的事件。

根據預設，每個事件匯流排最多可以設定 300 個規則。此配額可以在 Service Quotas 主控台中提高到數千個規則。由於規則限制適用於每個匯流排，因此如果您需要更多規則，您可以在帳戶中建立其他自訂事件匯流排。

您可以針對不同服務建立具有不同權限的事件匯流排，以自訂在帳戶中接收事件的方式。

若要在將事件 EventBridge 傳送至目標之前自訂事件的結構或日期，請使用輸入轉換器在進入目標之前編輯資訊。

如需詳細資訊，請參閱 ???。

目標

目標是一種資源或端點，當事件符合為規則定義的事件模式時，會將事件 EventBridge 傳送至該資源或端點。

目標可以從多個事件匯流排接收多個事件。

如需詳細資訊，請參閱 ???。

事件匯流排的進階功能

EventBridge 包含下列功能，可協助您開發、管理和使用事件匯流排。

使用 API 目的地啟用服務之間的 REST API 呼叫

EventBridge API 目標是您可以設定為規則目標的 HTTP 端點，方式與將事件資料傳送至 AWS 服務或資源的方式相同。透過使用 API 目的地，您可以使用 API 呼叫，在 AWS 服務、整合式 SaaS 應用程式和 AWS外部的應用程式之間路由事件。當您建立 API 目的地時，您可以指定要用於該目的地的連線。每個連線都包含授權類型的詳細資料，以及用於授權 API 目的地端點的參數。

封存和重播事件以協助開發和災難復原

您可以封存或儲存事件，然後稍後從封存中重新執行這些事件。封存可用於：

- 測試應用程式，因為您有要使用的事件存放區，而不必等待新事件。
- 注水一個新的服務，當它第一次上線。
- 為事件驅動的應用程式增加更多耐用性。

使用結構描述登錄檔來快速啟動事件模式建立

當您建置使用的無伺服器應用程式時 EventBridge，瞭解典型事件的結構可能會很有幫助，而不必產生事件。事件結構描述在結構描述中，這些結構描述適用於 AWS 服務在上產生的所有事件 EventBridge。

對於非來自 AWS 服務的事件，您可以：

- 建立或上傳自訂結構描述。

- 使用結構描述探索可為傳送至事件匯流排的事件 EventBridge 自動建立結構描述。

擁有事件的結構描述後，您就可以下載常用程式設計語言的程式碼繫結。

透過政策管理資源與存取

若要組織 AWS 資源或追蹤成本 EventBridge，您可以將自訂標籤或標籤指派給 AWS 資源。使用基於標籤的策略，您可以控制哪些資源可以在其中執行和不能執行 EventBridge。

除了以標籤為基礎的政策之外，還 EventBridge 支援以身分識別為基礎和以資源為基礎的政策來控制存取。 EventBridge使用以身分識別為基礎的政策來控制群組、角色或使用者的權限。使用以資源為基礎的政策為每個資源提供特定許可，例如 Lambda 函數或 Amazon SNS 主題。

創建一個 Amazon EventBridge 事件總線

您可以建立自訂事件匯流排，從您的自訂應用程式接收事件。您的應用程式也可以將事件傳送到您的預設事件匯流排。建立事件匯流排時，您可以附加以資源導向的政策，以授與其他帳戶的權限。然後，其他帳戶可以將事件發送到當前帳戶中的事件匯流排。

下面的視頻介紹如何建立事件匯流排：建立事件匯流排

建立自訂事件匯流排

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇事件匯流排。
3. 選擇 Create event bus (建立事件匯流排)。
4. 輸入新事件匯流排的名稱。
5. 設定事件匯流排：

   - 執行下列其中一項作業，指定以資源為基礎的策略：
     - 輸入包含要授與事件匯流排之權限的政策。您可以貼上來自其他來源的政策，或輸入該政策的 JSON。您可以使用其中一個範例原則，並針對您的環境進行修改。
     - 若要使用政策的範例，請選擇載入範例。根據您的環境適當修改政策，包括新增您授權政策中主體使用的其他動作。

     如需有關透過以資源為基礎的原則授與事件匯流排權限的詳細資訊，請參閱???。
   - 啟用歸檔（可選）

您可以建立事件歸檔，以便稍後輕鬆地重播事件。例如，您可能想要重播事件以從錯誤中還原或驗證應用程式中的新功能。如需更多資訊，請參閱 ???

a. 在「封存」下，選擇「啟用」

b. 指定歸檔的名稱和描述。

- 啟用結構描述探索 (選用)

  啟用結構描述探索，直接從此事件匯流排上執行的事件 EventBridge 自動推斷結構描述。如需更多資訊，請參閱 ???

  a. 在 [結構描述探索] 底下，選擇 [

- 指定標籤 (選擇性)

  標籤是您指派給 AWS 資源的自訂屬性標籤。使用標籤來識別和組織您的 AWS 資源。許多 AWS 服務都支援標記，因此您可以將相同標籤指派給來自不同服務的資源，以指出資源是相關的。如需更多資訊，請參閱 ???

  a. 在標籤下，選擇新增標籤。

  b. 指定新標籤的金鑰，並選擇性地指定值。

6. 選擇建立。

更新 Amazon EventBridge 事件總線

您可以在建立事件匯流排之後更新它們的組態。這包括默認事件總線，它會自動在您的帳戶中 EventBridge 創建。

主題

更新事件匯流排上的權限

您可以將以資源為基礎的政策附加至事件匯流排，將其他權限授與事件匯流排。如需有關更新指定事件匯流排之權限的詳細指示，請參閱管理事件匯流排權限。

在事件匯流排上新增或移除歸檔

歸檔可讓您擷取事件，以便日後輕鬆地重播事件。例如，您可能想要重播事件以從錯誤中還原或驗證應用程式中的新功能。如需詳細資訊，請參閱EventBridge 封存與重新顯示。

使用 EventBridge 主控台從事件匯流排新增或移除歸檔

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇事件匯流排。
3. 選擇您要更新的活動匯流排。
4. 在事件匯流排詳細資料頁面上，選擇 [封存] 索引標籤。
5. 執行以下任意一項：
   - 若要新增歸檔：
     a. 選擇建立封存。
     b. 指定歸檔的屬性。
     c. 選擇下一步。
     d. 選擇要套用至封存事件的事件模式。
     e. 選擇建立封存。
   - 若要刪除歸檔：
     a. 針對您要移除的標籤，選擇「刪除」。
     b. 輸入歸檔名稱，然後選擇「刪除」。

        歸檔將被永久刪除。您無法復原此操作。

若要使用建立或刪除事件匯流排的歸檔 AWS CLI

- 要創建歸檔，請使用創建歸檔。

  要永久刪除歸檔，請使用刪除歸檔。

啟動或停止事件匯流排上的結構描述探

如需結構描述探索的詳細資訊，請參閱結EventBridge 構描述

使用 EventBridge 主控台在事件匯流排上啟動或停止結構描述探索

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇事件匯流排。
3. 選擇您要更新的活動匯流排。
4. 執行以下任意一項：
   - 若要啟動結構描述探索，請選擇 [開始探索
   - 若要停止結構描述探索，請選擇刪除探查。

若要啟動或停止事件匯流排上的結構描述探索，請使用 AWS CLI

- 若要啟動結構描述探索，請使用建立探索器。

  若要停止結構描述探索，請使用刪除探索器。

在事件匯流排上新增或移除標籤

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。使用標籤來識別和組織您的 AWS 資源。如需詳細資訊，請參閱EventBridge 標籤。

使用 EventBridge 主控台從事件匯流排新增或移除標籤

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇事件匯流排。
3. 選擇您要更新的活動匯流排。
4. 在事件匯流排詳細資料頁面上，選擇 [標籤] 索引標籤，然後選擇 [管理標籤]。
5. 執行以下任意一項：
   - 若要新增標籤：
     a. 選擇 Add new tag (新增標籤)。
     b. 指定標籤的鍵和值
     c. 選擇更新。
   - 若要移除標籤：
     a. 針對您要移除的標籤，選擇「移除」。

b. 選擇更新。

若要使用在事件匯流排中新增或移除標籤 AWS CLI

- 要添加標籤，請使用標籤資源。

  要刪除標籤，請使用無標記資源。

刪除 Amazon EventBridge 事件總線

您可以刪除自訂或合作夥伴事件匯流排。您無法刪除預設事件匯流排。刪除事件匯流排會刪除與該事件匯流排相關聯的規則。

使用 EventBridge 主控台刪除事件匯流排

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇事件匯流排。
3. 選擇您要刪除的活動匯流排。
4. 執行以下任意一項：
   - 選擇 刪除。
   - 選擇活動匯流排的名稱。

     在事件匯流排詳細資訊頁面上，選擇 [刪除]。

Amazon EventBridge 事件匯流排的許可

您 AWS 帳戶中的預設事件匯流排只允許來自一個帳戶的事件。您可以將以資源為基礎的政策附加至事件匯流排，將其他權限授與事件匯流排。使用以資源為基礎的策略，您可以允許來自另一個帳戶的 `PutEvents`、`PutRule`、和 `PutTargets` API 呼叫。您也可以在政策中使用IAM 條件，將許可授予組織、套用標籤或僅篩選特定規則或帳戶中的事件。您可以在建立事件匯流排時或之後為事件匯流排設定以資源為基礎的政策。

接受事件匯流排 `Name` 參數 (例如 `PutRule`、`PutTargets`、`DeleteRule`、`RemoveTargets`、`DisableRule` 和 `EnableRule`) 的 EventBridge 各種API，也可以接受事件匯流排 ARN。使用這些參數可透過 API 參考跨帳戶或跨區域事件匯流排。例如，您可以呼叫 `PutRule` 在不同帳戶中的事件匯流排上建立規則，而不需要擔任角色。

您可以將本主題中的範例政策附加到 IAM 角色，以授予將事件傳送至其他帳戶或區域的權限。使用 IAM 角色設定組織控制政策和界限，瞭解誰可以將事件從您的帳戶傳送到其他帳戶。當規則的目標是事件匯流排時，建議一律使用 IAM 角色。您可以使用 `PutTarget` 呼叫附加 IAM 角色。如需關於建立規則以將事件傳送至其他帳戶或區域的相關資訊，請參閱 在 AWS 帳戶之間傳送和接收 Amazon EventBridge 事件。

主題

管理事件匯流排許可

使用下列程序修改現有事件匯流排的許可。如需關於如何使用 AWS CloudFormation 建立事件匯流排政策的詳細資訊，請參閱AWS:: Events:: EventBusPolicy。

管理現有事件匯流排的權限

1. 訪問 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇Event buses (事件匯流排)。
3. 在名稱中，選擇要管理其許可的事件匯流排名稱。

   如果資源策略已附加至事件匯流排，則會顯示策略。
4. 選擇管理許可，然後進行下列其中一個動作：
   - 輸入包含要授與事件匯流排之權限的政策。您可以貼上來自其他來源的政策，或輸入該政策的 JSON。
   - 若要使用政策的範例，請選擇載入範例。根據您的環境適當修改政策，包括新增您授權政策中主體使用的其他動作。
5. 選擇Update (更新)。

範本提供範例政策陳述式，您可以針對您的帳戶和環境自訂這些陳述式。範本不是有效的政策。您可以修改使用案例的範本，也可以複製其中一個範例政策並加以自訂。

範本會載入政策，其中包括如何授與帳戶以使用 PutEvents 動作的權限、如何授與權限給組織，以及如何授與帳戶管理帳戶規則的權限範例。您可以自訂特定帳戶的範本，然後從範本中刪除其他區段。本主題稍後將舉例說明。

如果您嘗試更新匯流排的權限，但政策包含錯誤，錯誤訊息會指出政策中的特定問題。

```
 ### Choose which sections to include in the policy to match your use case. ###
 ### Be sure to remove all lines that start with ###, including the ### at the end of
the line. ###

 ### The policy must include the following: ###

 {
   "Version": "2012-10-17",
   "Statement": [

     ### To grant permissions for an account to use the PutEvents action, include the
following, otherwise delete this section: ###

     {

       "Sid": "AllowAccountToPutEvents",
       "Effect": "Allow",
       "Principal": {
         "AWS": "<ACCOUNT_ID>"
       },
       "Action": "events:PutEvents",
       "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/default"
     },

     ### Include the following section to grant permissions to all members of your AWS
Organizations to use the PutEvents action ###

     {
       "Sid": "AllowAllAccountsFromOrganizationToPutEvents",
       "Effect": "Allow",
       "Principal": "*",
       "Action": "events:PutEvents",
       "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/default",
```

```
        "Condition": {
          "StringEquals": {
            "aws:PrincipalOrgID": "o-yourOrgID"
          }
        }
      },

      ### Include the following section to grant permissions to the account to manage
 the rules created in the account ###

      {
        "Sid": "AllowAccountToManageRulesTheyCreated",
        "Effect": "Allow",
        "Principal": {
          "AWS": "<ACCOUNT_ID>"
        },
        "Action": [
          "events:PutRule",
          "events:PutTargets",
          "events:DeleteRule",
          "events:RemoveTargets",
          "events:DisableRule",
          "events:EnableRule",
          "events:TagResource",
          "events:UntagResource",
          "events:DescribeRule",
          "events:ListTargetsByRule",
          "events:ListTagsForResource"],
        "Resource": "arn:aws:events:us-east-1:123456789012:rule/default",
        "Condition": {
          "StringEqualsIfExists": {
            "events:creatorAccount": "<ACCOUNT_ID>"
          }
        }
    }]
  }
```

範例政策：將事件傳送至不同帳戶中的預設匯流排

下列範例政策會授與帳戶 111122223333 的權限，以便將事件發佈至帳戶 123456789012 中的預設事件匯流排。

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [
       {
        "Sid": "sid1",
        "Effect": "Allow",
        "Principal": {"AWS":"arn:aws:iam::111112222333:root"},
        "Action": "events:PutEvents",
        "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/default"
        }
    ]
  }
```

範例政策：將事件傳送至不同帳戶中的預設匯流排

下列範例政策授與帳戶 111122223333 權限，可將事件發佈至帳戶 123456789012，但僅適用於來源值 central-event-bus 設定為 com.exampleCorp.webStore 和來源值 detail-type 設定為 newOrderCreated 的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WebStoreCrossAccountPublish",
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/central-event-bus",
      "Condition": {
        "StringEquals": {
          "events:detail-type": "newOrderCreated",
          "events:source": "com.exampleCorp.webStore"
        }
      }
    }
  ]
}
```

範例政策：將事件傳送至相同帳戶中的事件匯流排

下列附加至事件匯流排的範例政策 CustomBus1 可讓事件匯流排接收來自相同帳戶和區域的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "arn:aws:events:us-east-1:123456789:event-bus/CustomBus1"
      ]
    }
  ]
}
```

政策範例：將事件傳送至相同帳戶並限制更新

下列範例策略授與帳號 123456789012 權限，以建立、刪除、更新、停用和啟用規則，以及新增或移除目標。它會限制這些符合具有來源之事件的規則 com.exampleCorp.webStore，並使用 "events:creatorAccount": "${aws:PrincipalAccount}" 來確保只有帳戶 123456789012 可以在建立這些規則和目標之後修改這些規則和目標。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InvoiceProcessingRuleCreation",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": [
        "events:PutRule",
        "events:DeleteRule",
        "events:DescribeRule",
        "events:DisableRule",
        "events:EnableRule",
        "events:PutTargets",
```

```
        "events:RemoveTargets"
      ],
      "Resource": "arn:aws:events:us-east-1:123456789012:rule/central-event-bus/*",
      "Condition": {
        "StringEqualsIfExists": {
          "events:creatorAccount": "${aws:PrincipalAccount}",
          "events:source": "com.exampleCorp.webStore"
        }
      }
    }
  ]
}
```

範例政策：僅從特定規則傳送事件至不同區域的匯流排

下列範例政策授與帳戶 111122223333 權限，令其可將符合中東 (巴林) 和美國西部 (奧勒岡) 區域中以 SendToUSE1AnotherAccount 命名之規則的事件傳送至美國東部 (維吉尼亞北部) 帳戶 123456789012 中以 CrossRegionBus 命名的事件匯流排。範例政策會新增至帳戶 123456789012 CrossRegionBus 中指定的事件匯流排。只有當事件符合帳戶 111122223333 中為事件匯流排指定的規則時，此政策才允許事件。陳述式 Condition 會將事件限制為僅符合規則與指定規則 ARN 的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSpecificRulesAsCrossRegionSource",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:111112222333:rule/CrossRegionBus/
SendToUSE1AnotherAccount",
            "arn:aws:events:me-south-1:111112222333:rule/CrossRegionBus/
SendToUSE1AnotherAccount"
          ]
        }
```

```
      }
    }
  ]
}
```

範例政策：僅從一個特定區域傳送事件至不同區域

下列範例政策授與帳戶 111122223333 權限，令其可將在中東 (巴林) 和美國西部 (奧勒岡) 區域中發生的事件傳送至美國東部 (維吉尼亞北部) 帳戶 123456789012 中以 CrossRegionBus 命名的事件匯流排。帳戶 111122223333 沒有傳送任何其他區域產生之事件的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossRegionEventsFromUSWest2AndMESouth1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:*:*",
            "arn:aws:events:me-south-1:*:*"
          ]
        }
      }
    }
  ]
}
```

範例政策：拒絕從特定區域傳送事件

下列附加至帳戶 123456789012 中以 CrossRegionBus 命名的事件匯流排的範例政策會授與事件匯流排接收來自帳戶 111122223333 的事件的權限，但不授與在美國西部 (奧勒岡) 區域產生的事件。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "1AllowAnyEventsFromAccount111112222333",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus"
    },
    {
      "Sid": "2DenyAllCrossRegionUSWest2Events",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:*:*"
          ]
        }
      }
    }
  ]
}
```

從 Amazon EventBridge 事件匯流排產生 AWS CloudFormation 範本

AWS CloudFormation 透過將基礎架構視為程式碼，讓您以集中且可重複的方式，跨帳戶和區域設定和管理 AWS 資源。CloudFormation 透過讓您建立範本 (定義您要佈建和管理的資源) 來執行此作業。

EventBridge 可讓您從帳戶中現有的事件匯流排產生範本，以協助您快速開始開發 CloudFormation 範本。此外，EventBridge 還提供了在範本中包含與該事件匯流排相關聯的規則的選項。然後，您可以使用這些範本作為基礎，以便建立受 CloudFormation 管理的資源堆疊。

如需 CloudFormation 的詳細資訊，請參閱《AWS CloudFormation 使用者指南》。

> **Note**
>
> EventBridge 不會在產生的範本中包含受管理的規則。

您也可以從所選事件匯流排中包含的一或多個規則產生範本。

若要從事件匯流排產生 CloudFormation 範本

1. 訪問 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇事件匯流排。
3. 選擇您要從中產生 CloudFormation 範本的事件匯流排。
4. 從動作功能表中，選擇CloudFormation 範本，然後選擇您希望 EventBridge 在其中產生範本的格式：JSON 或 YAML。

   EventBridge 會顯示以所選格式產生的範本。依預設，與事件匯流排相關聯的所有規則都包含在樣板中。

   - 若要產生不包含規則的範本，請取消選取在此 EventBus 上包含規則。
5. EventBridge 可讓您選擇下載範本檔案，或將範本複製到剪貼簿。
   - 選擇立即下載以下載範本檔案。
   - 若要將範本複製剪貼簿，請選擇複製。
6. 若要結束範本，請選擇取消。

根據使用案例的需要自訂 AWS CloudFormation 範本後，您可以使用此範本在 CloudFormation 中建立堆疊。

使用從 Amazon EventBridge 產生的 CloudFormation 範本時的注意事項

使用從事件匯流排產生的 CloudFormation 範本時，請考量下列因素：

- EventBridge 不會在產生的範本中包含任何密碼。

  您可以編輯範本以包含範本參數，讓使用者在使用範本建立或更新 CloudFormation 堆疊時，能夠指定密碼或其他敏感資訊。

此外，使用者可以使用 Secrets Manager 在所需區域中建立密碼，然後編輯產生的範本以使用動態參數。

- 產生的範本中的目標會保持與原始事件匯流排中指定的完全相同。如果您在使用範本在其他地區建立堆疊之前未適當地編輯範本，這可能會導致跨區域問題。

  此外，產生的範本不會自動建立下游目標。

Amazon EventBridge 活動

事件指示某個環境 (例如 AWS 環境、SaaS 合作夥伴服務或應用程式) 或者您的某個應用程式或服務的變更。下列為事件的範例：

- 當執行個體的狀態從待定變更為執行中時，Amazon EC2 會產生一個事件。
- Amazon EC2 Auto Scaling 會在啟動或終止執行個體時產生事件。
- AWS CloudTrail 在您進行 API 呼叫時發佈事件。

您也可以設定定期產生的排程事件。

如需產生事件的服務清單，包括每個服務的範例事件，請參閱 來自 AWS 服務的事件 並追蹤表格中的連結。

事件表示為 JSON 物件，所有事件都具有類似的結構，且具有相同的最上層欄位。

detail (詳細資訊) 最上層欄位的內容各不相同，依據產生事件的服務及事件的內容而定。source (來源) 和 detail-type (詳細資訊類型) 欄位的組合用於識別欄位和 detail (詳細資訊) 欄位中的值。如需 AWS 服務產生之事件的範例，請參閱來自 AWS 服務的事件。

主題

- 事件結構參考
- 添加 Amazon EventBridge 事件 PutEvents
- 來自 AWS 服務的事件
- 接收來自 SaaS 合作夥伴與 Amazon 的事件 EventBridge
- 調試 Amazon EventBridge 事件交付

下列影片說明事件的基本概念：什麼是事件

下列影片說明活動到達的方式 EventBridge：事件來自何處

事件結構參考

事件中會顯示下列欄位：

```
{
  "version": "0",
  "id": "UUID",
  "detail-type": "event name",
  "source": "event source",
  "account": "ARN",
  "time": "timestamp",
  "region": "region",
  "resources": [
    "ARN"
  ],
  "detail": {
    JSON object
  }
}
```

version

根據預設，這在所有事件中皆設定為 0 (零)。

id

為每個事件產生的版本 4 UUID。您可以使用 `id` 追蹤事件從規則移至目標的過程。

詳細資訊類型

結合 source (來源) 欄位，識別顯示在 detail (詳細資訊) 欄位的欄位和值。

由傳遞的事件 CloudTrail 具有`AWS API Call via CloudTrail`作為的值`detail-type`。

source

識別產生事件的服務。所有來自 AWS 服務的事件都以“aws”開頭。客戶產生的事件在這裡可以有任何值，只要不以「aws」開頭。我們建議使用 Java 套件-名稱樣式的反向網域名稱字串。

若要尋找 AWS 服務的正確值，請參閱條件索引鍵資料表、從清單中選取服務，然後尋找服務前置詞。`source`例如，Amazon 的`source`價值 CloudFront 是`aws.cloudfront`。

帳戶

識別 AWS 帳戶的 12 位數字。

time

時間戳記，可由產生該事件的服務指定。如果事件涵蓋時間間隔，該服務會報告開始時間，因此該值可能早於接收到事件的時間。

region

識別事件起源的「 AWS 區域」。

resources

JSON 陣列包含 ARN，它可識別涉及該事件的資源。產生事件的服務會決定是否要包含這些 ARN。例如，Amazon EC2 執行個體狀態變更包含 Amazon EC2 執行個體 ARN，Auto Scaling 事件包含執行個體和 Auto Scaling 群組的 ARN，但 AWS CloudTrail 的 API 呼叫不包含資源 ARN。

詳細資訊

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。詳細內容可以像兩個字段一樣簡單。 AWS API 呼叫事件具有詳細資料物件，其中大約有 50 個欄位巢狀多層深。

Example 範例：Amazon EC2 執行個體狀態 - 變更通知

Amazon 的以下事件 EventBridge 表明亞 Amazon EC2 實例被終止。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ec2:us-west-1:123456789012:instance/i-1234567890abcdef0"
  ],
  "detail": {
    "instance-id": " i-1234567890abcdef0",
    "state": "terminated"
  }
}
```

有效自訂事件所需的最低資訊

建立自訂事件時，它們必須包含下列欄位：

```
{
  "detail-type": "event name",
```

```
  "source": "event source",
  "detail": {
  }
}
```

- `detail`：包含事件相關資訊的 JSON 物件。它可以是 "{}"。

> **Note**
>
> PutEvents接受 JSON 格式的資料。針對 JSON 數字 (整數) 資料類型，條件約束為：最小值為 -9,223,372,036,854,775,808，最大值為 9,223,372,036,854,775,807。

- `detail-type`：識別事件類型的字串。
- `source`：識別事件來源的字串。客戶產生的事件在這裡可以有任何值，只要不以「aws」開頭。我們建議使用 Java 套件-名稱樣式的反向網域名稱字串。

添加 Amazon EventBridge 事件 **PutEvents**

`PutEvents`動作會在單一要求 EventBridge 中將多個事件傳送至。如需詳細資訊，請參閱 PutEventsAmazon EventBridge API 參考和AWS CLI 命令參考中的放置事件。

每個 `PutEvents` 請求可支援有限數量的項目。如需詳細資訊，請參閱 Amazon EventBridge 配額。`PutEvents` 操作嘗試依照請求的自然順序處理所有項目。呼叫之後`PutEvents`， EventBridge 為每個事件指派一個唯一的 ID。

主題

- 使用 PutEvents 處理失敗
- 使用傳送事件 AWS CLI
- 計算 Amazon EventBridge PutEvents 事件條目大小

下列範例 Java 程式碼會將兩個相同的事件傳送至 EventBridge。

AWS SDK for Java Version 2.x

```
EventBridgeClient eventBridgeClient =
    EventBridgeClient.builder().build();

PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
```

```
    .resources("resource1", "resource2")
    .source("com.mycompany.myapp")
    .detailType("myDetailType")
    .detail("{ \"key1\": \"value1\", \"key2\": \"value2\" }")
    .build();

List <
PutEventsRequestEntry > requestEntries = new ArrayList <
PutEventsRequestEntry > ();
requestEntries.add(requestEntry);

PutEventsRequest eventsRequest = PutEventsRequest.builder()
    .entries(requestEntries)
    .build();

PutEventsResponse result = eventBridgeClient.putEvents(eventsRequest);

for (PutEventsResultEntry resultEntry: result.entries()) {
    if (resultEntry.eventId() != null) {
        System.out.println("Event Id: " + resultEntry.eventId());
    } else {
        System.out.println("PutEvents failed with Error Code: " +
 resultEntry.errorCode());
    }
}
```

AWS SDK for Java Version 1.0

```
EventBridgeClient eventBridgeClient =
    EventBridgeClient.builder().build();

PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
        .withTime(new Date())
        .withSource("com.mycompany.myapp")
        .withDetailType("myDetailType")
        .withResources("resource1", "resource2")
        .withDetail("{ \"key1\": \"value1\", \"key2\": \"value2\" }");

PutEventsRequest request = new PutEventsRequest()
        .withEntries(requestEntry, requestEntry);

PutEventsResult result = awsEventsClient.putEvents(request);
```

```
for (PutEventsResultEntry resultEntry : result.getEntries()) {
    if (resultEntry.getEventId() != null) {
        System.out.println("Event Id: " + resultEntry.getEventId());
    } else {
        System.out.println("Injection failed with Error Code: " +
 resultEntry.getErrorCode());
    }
}
```

執行此程式碼之後，PutEvents 結果會包含回應項目的陣列。回應陣列中的每個項目按照從請求和回應的開始到結束的順序對應於請求陣列中的一個項目。回應 Entries 陣列通常包含與請求陣列相同數量的項目。

使用 PutEvents 處理失敗

根據預設，如果要求中的個別項目失敗，會 EventBridge 繼續處理要求中其餘的項目。回應 Entries 陣列可以同時包含成功和失敗的項目。您必須偵測未成功處理的項目，並將它們包含於後續呼叫中。

成功的結果項目包含一個 Id 值，不成功的結果項目包括 ErrorCode 和 ErrorMessage 值。ErrorCode 描述錯誤的類型。ErrorMessage 提供關於錯誤的詳細資訊。以下範例有 PutEvents 請求的三個結果項目。第二個項目不成功。

```
{
    "FailedEntryCount": 1,
    "Entries": [
        {
            "EventId": "11710aed-b79e-4468-a20b-bb3c0c3b4860"
        },
        {   "ErrorCode": "InternalFailure",
            "ErrorMessage": "Internal Service Failure"
        },
        {
            "EventId": "d804d26a-88db-4b66-9eaf-9a11c708ae82"
        }
    ]
}
```

> **Note**
>
> 如果您使用將事件發佈PutEvents至不存在的事件匯流排，則 EventBridge 事件比對將不會找到對應的規則，而且會捨棄該事件。雖然 EventBridge 會發送200響應，但它不會失敗請求或將事件包含在請求響應的FailedEntryCount值中。

未成功處理的項目可包含在後續的 PutEvents 請求中。首先，查看 PutEventsResult 中的 FailedRecordCount 參數以確認請求中是否有失敗的項目。如果其值非為零，您可以將具有非 null 值之 ErrorCode 中的每個 Entry 新增至後續請求中。以下範例顯示一個失敗處理常式。

```
PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
        .withTime(new Date())
        .withSource("com.mycompany.myapp")
        .withDetailType("myDetailType")
        .withResources("resource1", "resource2")
        .withDetail("{ \"key1\": \"value1\", \"key2\": \"value2\" }");

List<PutEventsRequestEntry> putEventsRequestEntryList = new ArrayList<>();
for (int i = 0; i < 3; i++) {
    putEventsRequestEntryList.add(requestEntry);
}

PutEventsRequest putEventsRequest = new PutEventsRequest();
putEventsRequest.withEntries(putEventsRequestEntryList);
PutEventsResult putEventsResult = awsEventsClient.putEvents(putEventsRequest);

while (putEventsResult.getFailedEntryCount() > 0) {
    final List<PutEventsRequestEntry> failedEntriesList = new ArrayList<>();
    final List<PutEventsResultEntry> PutEventsResultEntryList =
 putEventsResult.getEntries();
    for (int i = 0; i < PutEventsResultEntryList.size(); i++) {
        final PutEventsRequestEntry putEventsRequestEntry =
 putEventsRequestEntryList.get(i);
        final PutEventsResultEntry putEventsResultEntry =
 PutEventsResultEntryList.get(i);
        if (putEventsResultEntry.getErrorCode() != null) {
            failedEntriesList.add(putEventsRequestEntry);
        }
    }
    putEventsRequestEntryList = failedEntriesList;
    putEventsRequest.setEntries(putEventsRequestEntryList);
```

```
    putEventsResult = awsEventsClient.putEvents(putEventsRequest);
    }
```

使用傳送事件 AWS CLI

您可以使用將自訂事件傳送 AWS CLI 至，以 EventBridge 便處理這些事件。下列範例會將一個自訂事件放入 EventBridge：

```
aws events put-events \
--entries '[{"Time": "2016-01-14T01:02:03Z", "Source": "com.mycompany.myapp",
 "Resources": ["resource1", "resource2"], "DetailType": "myDetailType", "Detail":
 "{ \"key1\": \"value1\", \"key2\": \"value2\" }"}]'
```

您也可以建立包含自訂事件的 JSON 檔案。

```
[
  {
    "Time": "2016-01-14T01:02:03Z",
    "Source": "com.mycompany.myapp",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType",
    "Detail": "{ \"key1\": \"value1\", \"key2\": \"value2\" }"
  }
]
```

然後，若要使用讀取此檔案中的項目並傳送事件，請在命令提示字元中輸入： AWS CLI

```
aws events put-events --entries file://entries.json
```

計算 Amazon EventBridge PutEvents 事件條目大小

您可以使用PutEvents動作將 EventBridge 自訂事件傳送至。您可以將多個事件項目以批次方式加入一個請求，以提高效率。項目總大小必須少於 256KB。您可以在傳送事件之前計算項目大小。

> **Note**
>
> 項目有大小的限制。即使項目小於大小限制，由於事件的 EventBridge JSON 表示法的必要字元和索引鍵，因此中的事件永遠大於項目大小。如需詳細資訊，請參閱 Amazon EventBridge 活動。

EventBridge 計算大PutEventsRequestEntry小，如下所示：

- 如果已指定，Time 參數為 14 個位元。
- Source 和 DetailType 參數為其 UTF-8 編碼表單的位元組數。
- 如果已指定，Detail 參數為其 UTF-8 編碼表單的位元組數。
- 如果已指定，Resources 參數的每個項目為其 UTF-8 編碼表單的位元組數。

以下範例 Java 程式碼會計算指定的 PutEventsRequestEntry 物件的大小：

```
int getSize(PutEventsRequestEntry entry) {
    int size = 0;
    if (entry.getTime() != null) {
        size += 14;
    }
    size += entry.getSource().getBytes(StandardCharsets.UTF_8).length;
    size += entry.getDetailType().getBytes(StandardCharsets.UTF_8).length;
    if (entry.getDetail() != null) {
        size += entry.getDetail().getBytes(StandardCharsets.UTF_8).length;
    }
    if (entry.getResources() != null) {
        for (String resource : entry.getResources()) {
            if (resource != null) {
                size += resource.getBytes(StandardCharsets.UTF_8).length;
            }
        }
    }
    return size;
```

```
}
```

> **Note**
>
> 如果項目大小大於 256KB，我們建議將事件上傳至 Amazon S3 儲存貯體，並在 `PutEvents` 項目中加入 `Object URL`。

來自 AWS 服務的事件

許多 AWS 服務會產生 EventBridge 接收的事件。當您帳戶中的某個 AWS 服務發出事件時，它會移至您帳戶的預設事件匯流排。

從 AWS 服務交付事件

每個產生事件的 AWS 服務都會以最佳方 EventBridge 式或持久的傳遞嘗試傳送給這些事件。

- 盡力傳遞意味著服務會嘗試將所有事件傳送至 EventBridge，但在極少數情況下，事件可能無法傳遞。
- 持久交付意味著服務將成功嘗試將事件傳遞給至 EventBridge 少一次。

  EventBridge 將在正常情況下接受所有有效事件。如果由於 EventBridge 服務中斷而無法傳遞事件，則服 AWS 務稍後將重試最多 24 小時。

一旦事件傳遞到 EventBridge，就會與規則進行 EventBridge 比對，然後遵循重試原則以及為事件目標指定的任何無效字母佇列。

如需產生事件的 AWS 服務清單，請參閱???。

透過存取 AWS 服務事件 AWS CloudTrail

AWS CloudTrail 是自動記錄事件 (例如 AWS API 呼叫) 的服務。您可以建立使用來源資訊的 EventBridge 規則 CloudTrail。如需有關的詳細資訊 CloudTrail，請參閱什麼是 AWS CloudTrail？ 。

由傳送的所有事件都 CloudTrail 具`AWS API Call via CloudTrail`有的值`detail-type`。

若要記錄`detail-type`值為的事件`AWS API Call via CloudTrail`，需要啟用記錄功能的 CloudTrail 追蹤。

CloudTrail 搭配 Amazon S3 使用時，您需要進行設定 CloudTrail 以記錄資料事件。如需詳細資訊，請參閱啟用 S3 儲存貯體和物件的 CloudTrail 事件記錄。

AWS 服務中的某些事件可以由服務本身報告，也可以由服務報告 CloudTrail。 EventBridge 例如，啟動或停止執行個體的 Amazon EC2 API 呼叫會透過以下方式產生 EventBridge 事件和事件 CloudTrail。

CloudTrail 支援 API 呼叫者和資源擁有者透過建立追蹤來接收 Amazon S3 儲存貯體中的事件，並透過將事件傳遞給 API 呼叫者。 EventBridge除了 API 呼叫者之外，資源擁有者還可以透過以下方式監控跨帳戶 API 呼叫。 EventBridge CloudTrail與的整合 EventBridge 提供了一種便利的方式來設定自動化規則型工作流程以回應事件。

您無法使用大於 256 KB 的 AWS Put* 事件 API 呼叫事件作為事件模式，因為任何 Put* 事件要求的大小上限為 256 KB。如需有關可使用之 API 呼叫的詳細資訊，請參閱CloudTrail 支援的服務和整合。

從 AWS 服務接收唯讀管理事件

您可以在預設或自訂事件匯流排上設定規則，以透過以下方式接收來自 AWS 服務的唯讀管理事件 CloudTrail。管理事件可讓您了解對 AWS 帳戶中的資源執行的管理作業。這些也稱為控制平面操作。如需詳細資訊，請參閱《CloudTrail 使用者指南》中的記錄管理事件。

針對有關預設或自訂事件匯流排的每個規則，您可以設定規則狀態來控制要接收的事件類型：

- 停用規則，使其 EventBridge 不符合規則的事件。
- 啟用規則，使事件與規則 EventBridge 相符，但透過傳送的唯讀 AWS 管理事件除外 CloudTrail。
- 啟用規則，使所有事件與規則 EventBridge 相符，包括透過傳遞的唯讀管理事件 CloudTrail。

合作夥伴活動匯流排不會接收 AWS 事件。

決定是否接收唯讀管理事件時需要考量的一些事項：

- 某些唯讀管理事件 (例如 AWS Key Management Service `GetKeyPolicy`和`DescribeKey`或 IAM `GetPolicy` 和`GetRole`事件) 發生在比典型變更事件高得多的磁碟區。
- 如果這些事件不是以 `Describe`、`Get` 或 `List` 開頭，您可能已經收到唯讀的管理事件。例如，來自以下 AWS STS API 的事件是變更事件，甚至認為它們以動詞開頭`Get`：
  - `GetFederationToken`
  - `GetSessionToken`

  如需不遵守`Describe`、`Get`或`List`命名慣例 (依 AWS 服務) 的唯讀管理事件清單，請參閱???。

建立使用 AWS CLI 接收唯讀管理事件的規則

- 使用 put-rule 指令建立或更新規則，並使用參數執行下列作業：
  - 指定規則屬於預設事件匯流排或特定自訂事件匯流排
  - 將規則狀態設為 ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS

```
aws events put-rule --name "ruleForManagementEvents" --event-bus-name
"default" --state "ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS"
```

> **Note**
>
> 僅透過 AWS CLI 和 AWS CloudFormation 範本支援針對 CloudWatch 管理事件啟用規則。

Example

下列範例說明如何與特定事件進行比對。最佳實務是為比對特定事件定義專用規則，以保持清晰和易於編輯。

在此情況下，專用規則會符合來源的AssumeRole管理事件 AWS Security Token Service。

```
{
    "source" : [ "aws.sts" ],
    "detail-type": ["AWS API Call via CloudTrail"],
    "detail" : {
        "eventName" : ["AssumeRole"]
    }
}
```

AWS 產生事件的服務

下表顯示產生事件的 AWS 服務。選擇服務名稱，即可查看有關該服務和如何協同 EventBridge 運作的詳細資訊。

每個產生事件的 AWS 服務都會以最佳方 EventBridge 式或持久的傳遞嘗試傳送給這些事件。如需詳細資訊，請參閱 ???。

此表格包含將事件傳送至的 AWS 服務的表示法 EventBridge，但並不包含所有服務。對於未列出事件的服務 EventBridge，請假設盡最大努力傳遞。

服務	嘗試類型
企業版 Alexa	全力
AWS Account Management	全力
Amazon API Gateway	全力
AWS AppConfig	全力
Amazon AppFlow	全力
Application Auto Scaling	全力
AWS 應用程式成本分析工具	全力
AWS Application Migration Service	全力
Amazon Athena	全力
AWS Backup	全力
AWS Batch	耐用
Amazon Braket	耐用
AWS Certificate Manager	全力
Amazon Chime	全力
Amazon 雲端目錄	全力
AWS CloudFormation	耐用
Amazon CloudFront	全力
AWS CloudHSM	全力
Amazon CloudSearch	全力
AWS CloudShell	全力

服務	嘗試類型
從事件 AWS CloudTrail	全力
Amazon CloudWatch	耐用
Amazon CloudWatch 應用洞察	全力
Amazon CloudWatch 網絡監控	全力
Amazon CloudWatch 日誌	全力
Amazon CloudWatch Synthetics	全力
AWS CodeArtifact	耐用
AWS CodeBuild	全力
AWS CodeCommit	全力
AWS CodeDeploy	全力
Amazon CodeGuru 分析器	全力
AWS CodePipeline	全力
AWS CodeStar	全力
CodeConnections	全力
Amazon Cognito 身分	全力
Amazon Cognito 使用者集區	全力
Amazon Cognito Sync	全力
AWS Config	全力
Amazon Connect	全力
Amazon Connect Voice ID	全力

服務	嘗試類型
AWS Control Tower	全力
AWS Database Migration Service	全力
AWS Data Exchange	全力
Amazon Data Lifecycle Manager	全力
AWS Data Pipeline	全力
AWS DataSync	全力
AWS Device Farm	全力
Amazon DevOps 大師	全力
AWS Direct Connect	全力
AWS Directory Service	全力
Amazon DynamoDB	全力
AWS Elastic Beanstalk	全力
Amazon Elastic Block Store	全力
Amazon Elastic Block Store 磁碟區修改	全力
Amazon ElastiCache	全力
Amazon Elastic Compute Cloud (Amazon EC2)	全力
Amazon EC2 Auto Scaling	全力
Amazon EC2 機群	全力
Amazon EC2 Spot 執行個體中斷事件	全力
Amazon Elastic Container Registry	全力

服務	嘗試類型
Amazon Elastic Container Service	耐用
AWS Elastic Disaster Recovery	全力
Amazon Elastic File System	全力
Amazon Elastic Kubernetes Service	全力
Elastic Load Balancing	全力
Amazon 彈性 MapReduce	全力
Amazon Elastic Transcoder	全力
AWS Elemental MediaConnect	全力
AWS Elemental MediaConvert	耐用
AWS Elemental MediaLive	全力
AWS Elemental MediaPackage	全力
AWS Elemental MediaStore	耐用
Amazon EMR	全力
Amazon EMR on EKS	全力
Amazon EMR Serverless	全力
Amazon EventBridge 排程規則	耐用
Amazon EventBridge 模式	全力
AWS Fault Injection Service	全力
預測	全力
Amazon GameLift	全力

服務	嘗試類型
AWS Glue	全力
AWS Glue DataBrew	全力
AWS Ground Station	全力
Amazon GuardDuty	全力
AWS Health	耐用
AWS HealthLake	耐用
AWS Identity and Access Management (IAM)	全力
IAM Access Analyzer	全力
Amazon Inspector Classic	全力
Amazon Inspector	全力
AWS IoT	全力
AWS IoT Analytics	耐用
AWS IoT Greengrass V1	全力
AWS IoT Greengrass V2	全力
Amazon Interactive Video Service	全力
Amazon Kinesis	全力
Amazon 數據 Firehose	全力
AWS Key Management Service 刪除 CMK	耐用
AWS Key Management Service 軸向旋轉	全力
AWS Key Management Service 匯入的金鑰材料過期	全力

服務	嘗試類型
AWS Lambda	全力
Amazon Location Service	耐用
Amazon Machine Learning	全力
Amazon Macie	全力
Amazon Managed Blockchain	全力
AWS Managed Services	全力
AWS Management Console 登入	全力
AWS 計量 Marketplace	全力
AWS Migration Hub	全力
AWS Migration Hub Refactor Spaces	全力
AWS 監控	全力
AWS Network Manager	全力
Amazon OpenSearch 服務	全力
AWS OpsWorks	耐用
AWS OpsWorks CM	全力
AWS Organizations	全力
Amazon Polly	全力
AWS Private Certificate Authority	全力
AWS Proton	全力
Amazon QLDB	耐用

服務	嘗試類型
Amazon QuickSight	全力
Amazon RDS	全力
AWS 資源回收筒	全力
Amazon Redshift	耐用
Amazon Redshift 資料 API	全力
Amazon Redshift Serverless	全力
AWS Resource Access Manager	全力
AWS Resource Groups	全力
AWS Resource Groups Tagging API	全力
Amazon Route 53	全力
Amazon Route 53 Recovery Readiness	全力
Amazon SageMaker	全力
Savings Plans	全力
AWS Secrets Manager	全力
AWS Security Hub	耐用
AWS Security Token Service	全力
AWS Server Migration Service	全力
AWS Service Catalog	全力
AWS Signer	耐用
Amazon Simple Email Service	全力

服務	嘗試類型
Amazon Simple Storage Service (Amazon S3)	耐用
Amazon S3 Glacier	全力
Amazon S3 on Outposts	全力
Amazon Simple Queue Service	全力
Amazon Simple Notification Service	全力
Amazon Simple Workflow Service	全力
AWS Step Functions	全力
AWS Storage Gateway	耐用
AWS Support	全力
AWS Systems Manager	全力
Amazon Transcribe	全力
AWS Transfer Family	全力
AWS Transit Gateway	全力
Amazon Translate	耐用
AWS Trusted Advisor	全力
AWS WAF	全力
AWS WAF 區域	全力
AWS Well-Architected Tool	全力
Amazon WorkDocs	全力
Amazon WorkSpaces	全力

服務	嘗試類型
AWS X-Ray	全力

AWS 服務產生的管理事件

一般而言，產生管理 (或唯讀) 事件的 API 會以動詞 `Describe`、`Get` 或 `List` 開頭。下表列出不遵循此命名慣例的 AWS 服務及其產生的管理事件。如需管理事件的詳細資訊，請參閱 ???。

不是以 **`Describe`**、**`Get`** 或 **`List`** 開頭的管理事件

下表列出 AWS 服務及其所產生的管理事件，這些服務與管理事件不遵循開頭為`Describe`、`Get`、或的一般命名慣例`List`。

服務	事件名稱	事件類型
企業版 Alexa	ResolveRoom	API 呼叫
企業版 Alexa	SearchAddressBooks	API 呼叫
企業版 Alexa	SearchContacts	API 呼叫
企業版 Alexa	SearchDevices	API 呼叫
企業版 Alexa	SearchProfiles	API 呼叫
企業版 Alexa	SearchRooms	API 呼叫
企業版 Alexa	SearchSkillGroups	API 呼叫
企業版 Alexa	SearchUsers	API 呼叫
IAM Access Analyzer	ValidatePolicy	API 呼叫
AWS AdSpace 潔淨的房間	BatchGetSchema	API 呼叫
AWS Amplify UI 生成器	ExportComponents	API 呼叫
AWS Amplify UI 生成器	ExportForms	API 呼叫

服務	事件名稱	事件類型
AWS Amplify UI 生成器	ExportThemes	API 呼叫
Amazon OpenSearch 服務	BatchGetCollection	API 呼叫
Amazon API Gateway	ExportApi	API 呼叫
AWS AppConfig	ValidateConfiguration	API 呼叫
Amazon AppFlow	RetrieveConnectorData	API 呼叫
Amazon CloudWatch 應用洞察	UpdateApplicationDashboardConfiguration	API 呼叫
Amazon Athena	BatchGetNamedQuery	API 呼叫
Amazon Athena	BatchGetPreparedStatement	API 呼叫
Amazon Athena	BatchGetQueryExecution	API 呼叫
Amazon Athena	CheckQueryCompatibility	API 呼叫
Amazon Athena	ExportNotebook	API 呼叫
AWS Auto Scaling	AreScalableTargetsRegistered	API 呼叫
AWS Auto Scaling	測試	API 呼叫
AWS Marketplace	SearchAgreements	API 呼叫
AWS Backup	CreateLegalHold	API 呼叫
AWS Backup	ExportBackupPlanTemplate	API 呼叫
AWS Backup gateway	TestHypervisorConfiguration	API 呼叫
AWS Billing and Cost Management	AWSPaymentInstrumentGateway獲取。	主控台動作

服務	事件名稱	事件類型
AWS Billing and Cost Management	AWSPaymentPortalService.DescribeMakePaymentPage	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.DescribePaymentsDashboard	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetAccountPreferences	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetAdvancePaySummary	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetAsoBulkDownload	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetBillingContactAddress	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetDocuments	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetEligiblePaymentInstruments	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetEntitiesByIds	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetFundingDocuments	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetKybcValidationStatus	主控台動作

服務	事件名稱	事件類型
AWS Billing and Cost Management	AWSPaymentPortalService.GetOneTimePasswordStatus	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentHistory	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileByArn	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileCurrencies	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfiles	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileServiceProviders	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentsDue	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetRemittanceInformation	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetTaxInvoiceMetadata	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetTermsAndConditionsForProgramGroup	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetTransactionsHistory	主控台動作

服務	事件名稱	事件類型
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnappliedFunds	主控台動作
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnpaidInvoices	主控台動作
AWS Billing and Cost Management	AWSPaymentPreferenceGateway獲取。	主控台動作
AWS Billing and Cost Management	CancelBulkDownload	主控台動作
AWS Billing and Cost Management	DownloadCommercialInvoice	主控台動作
AWS Billing and Cost Management	DownloadCsv	主控台動作
AWS Billing and Cost Management	DownloadDoc	主控台動作
AWS Billing and Cost Management	已下載的 CSV 檔案 ForBillingPeriod	主控台動作
AWS Billing and Cost Management	DownloadPaymentHistory	主控台動作
AWS Billing and Cost Management	DownloadRegistrationDocument	主控台動作
AWS Billing and Cost Management	DownloadTaxInvoice	主控台動作
AWS Billing and Cost Management	FindBankRedirectPaymentInstruments	主控台動作
AWS Billing and Cost Management	FindeSV ForBillingPeriod	主控台動作

服務	事件名稱	事件類型
AWS Billing and Cost Management	ValidateReportDestination	主控台動作
AWS Billing and Cost Management	VerifyChinaPaymentEligibility	主控台動作
Amazon Braket	SearchCompilations	API 呼叫
Amazon Braket	SearchDevices	API 呼叫
Amazon Braket	SearchQuantumTasks	API 呼叫
Amazon Connect Cases	BatchGetField	API 呼叫
Amazon Connect Cases	SearchCases	API 呼叫
Amazon Connect Cases	SearchRelatedItems	API 呼叫
Amazon Chime	RetrieveDataExports	API 呼叫
Amazon Chime	SearchChannels	API 呼叫
Amazon Chime SDK 身分	DeleteProfile	服務事件
Amazon Chime SDK 身分	DeleteWorkTalkAccount	服務事件
AWS 潔淨室	BatchGetSchema	API 呼叫
Amazon 雲端目錄	BatchRead	API 呼叫
Amazon 雲端目錄	LookupPolicy	API 呼叫
AWS CloudFormation	DetectStackDrift	API 呼叫
AWS CloudFormation	DetectStackResourceDrift	API 呼叫
AWS CloudFormation	DetectStackSetDrift	API 呼叫
AWS CloudFormation	EstimateTemplateCost	API 呼叫

服務	事件名稱	事件類型
AWS CloudFormation	ValidateTemplate	API 呼叫
AWS CloudShell	RedeemCode	API 呼叫
AWS CloudTrail	LookupEvents	API 呼叫
AWS CodeArtifact	ReadFromRepository	API 呼叫
AWS CodeArtifact	SearchPackages	API 呼叫
AWS CodeArtifact	VerifyResourcesExistForTagris	API 呼叫
AWS CodeBuild	BatchGetBuildBatches	API 呼叫
AWS CodeBuild	BatchGetBuilds	API 呼叫
AWS CodeBuild	BatchGetProjects	API 呼叫
AWS CodeBuild	BatchGetReportGroups	API 呼叫
AWS CodeBuild	BatchGetReports	API 呼叫
AWS CodeBuild	BatchPutCodeCoverages	API 呼叫
AWS CodeBuild	BatchPutTestCases	API 呼叫
AWS CodeBuild	RequestBadge	服務事件
AWS CodeCommit	BatchDescribeMergeConflicts	API 呼叫
AWS CodeCommit	BatchGetCommits	API 呼叫
AWS CodeCommit	BatchGetPullRequests	API 呼叫
AWS CodeCommit	BatchGetRepositories	API 呼叫
AWS CodeCommit	EvaluatePullRequestApprovalRules	API 呼叫

服務	事件名稱	事件類型
AWS CodeCommit	GitPull	API 呼叫
AWS CodeDeploy	BatchGetApplicationRevisions	API 呼叫
AWS CodeDeploy	BatchGetApplications	API 呼叫
AWS CodeDeploy	BatchGetDeploymentGroups	API 呼叫
AWS CodeDeploy	BatchGetDeployment Instances	API 呼叫
AWS CodeDeploy	BatchGetDeployments	API 呼叫
AWS CodeDeploy	BatchGetDeploymentTargets	API 呼叫
AWS CodeDeploy	BatchGetOnPremises Instances	API 呼叫
Amazon CodeGuru 分析器	BatchGetFrameMetricData	API 呼叫
Amazon CodeGuru 分析器	SubmitFeedback	API 呼叫
AWS CodePipeline	PollForJobs	API 呼叫
AWS CodePipeline	PollForThirdPartyJobs	API 呼叫
CodeConnections	StartAppRegistrationHandsha ke	API 呼叫
CodeConnections	斯塔圖 AuthHandshake	API 呼叫
CodeConnections	ValidateHostWebhook	API 呼叫
Amazon CodeWhisperer	CreateCodeScan	API 呼叫
Amazon CodeWhisperer	CreateProfile	API 呼叫
Amazon CodeWhisperer	CreateUploadUrl	API 呼叫
Amazon CodeWhisperer	GenerateRecommendations	API 呼叫

服務	事件名稱	事件類型
Amazon CodeWhisperer	UpdateProfile	API 呼叫
Amazon Cognito 身分	LookupDeveloperIdentity	API 呼叫
Amazon Cognito 使用者集區	AdminGetDevice	API 呼叫
Amazon Cognito 使用者集區	AdminGetUser	API 呼叫
Amazon Cognito 使用者集區	AdminListDevices	API 呼叫
Amazon Cognito 使用者集區	AdminListGroupsForUser	API 呼叫
Amazon Cognito 使用者集區	AdminListUserAuthEvents	API 呼叫
Amazon Cognito 使用者集區	Beta_Authorize_GET	服務事件
Amazon Cognito 使用者集區	Confirm_GET	服務事件
Amazon Cognito 使用者集區	ConfirmForgotPassword_ 取得	服務事件
Amazon Cognito 使用者集區	Error_GET	服務事件
Amazon Cognito 使用者集區	ForgotPassword_ 取得	服務事件
Amazon Cognito 使用者集區	IntrospectToken	API 呼叫
Amazon Cognito 使用者集區	Login_Error_POST	服務事件
Amazon Cognito 使用者集區	Login_GET	服務事件
Amazon Cognito 使用者集區	Mfa_GET	服務事件
Amazon Cognito 使用者集區	MfaOption_ 取得	服務事件
Amazon Cognito 使用者集區	ResetPassword_ 取得	服務事件
Amazon Cognito 使用者集區	Signup_GET	服務事件
Amazon Cognito 使用者集區	UserInfo_ 取得	服務事件

服務	事件名稱	事件類型
Amazon Cognito 使用者集區	UserInfo_ 帖子	服務事件
Amazon Cognito Sync	BulkPublish	API 呼叫
Amazon Comprehend	BatchContainsPiiEntities	API 呼叫
Amazon Comprehend	BatchDetectDominantLanguage	API 呼叫
Amazon Comprehend	BatchDetectEntities	API 呼叫
Amazon Comprehend	BatchDetectKeyPhrases	API 呼叫
Amazon Comprehend	BatchDetectPiiEntities	API 呼叫
Amazon Comprehend	BatchDetectSentiment	API 呼叫
Amazon Comprehend	BatchDetectSyntax	API 呼叫
Amazon Comprehend	BatchDetectTargetedSentiment	API 呼叫
Amazon Comprehend	ClassifyDocument	API 呼叫
Amazon Comprehend	ContainsPiiEntities	API 呼叫
Amazon Comprehend	DetectDominantLanguage	API 呼叫
Amazon Comprehend	DetectEntities	API 呼叫
Amazon Comprehend	DetectKeyPhrases	API 呼叫
Amazon Comprehend	DetectPiiEntities	API 呼叫
Amazon Comprehend	DetectSentiment	API 呼叫
Amazon Comprehend	DetectSyntax	API 呼叫
Amazon Comprehend	DetectTargetedSentiment	API 呼叫

服務	事件名稱	事件類型
Amazon Comprehend	DetectToxicContent	API 呼叫
AWS Compute Optimizer	ExportAutoScalingGroupRecommendations	API 呼叫
AWS Compute Optimizer	出口 VolumeRecommendations	API 呼叫
AWS Compute Optimizer	出口技術 InstanceRecommendations	API 呼叫
AWS Compute Optimizer	出口商 ServiceRecommendations	API 呼叫
AWS Compute Optimizer	ExportLambdaFunctionRecommendations	API 呼叫
AWS Compute Optimizer	出口商 InstanceRecommendations	API 呼叫
AWS Config	BatchGetAggregateResourceConfig	API 呼叫
AWS Config	BatchGetResourceConfig	API 呼叫
AWS Config	SelectAggregateResourceConfig	API 呼叫
AWS Config	SelectResourceConfig	API 呼叫
Amazon Connect	AdminGetEmergencyAccessToken	API 呼叫
Amazon Connect	SearchQueues	API 呼叫
Amazon Connect	SearchRoutingProfiles	API 呼叫
Amazon Connect	SearchSecurityProfiles	API 呼叫

服務	事件名稱	事件類型
Amazon Connect	SearchUsers	API 呼叫
AWS Glue DataBrew	SendProjectSessionAction	API 呼叫
AWS Data Pipeline	EvaluateExpression	API 呼叫
AWS Data Pipeline	QueryObjects	API 呼叫
AWS Data Pipeline	ValidatePipelineDefinition	API 呼叫
AWS DataSync	VerifyResourcesExistForTagris	API 呼叫
AWS DeepLens	BatchGetDevice	API 呼叫
AWS DeepLens	BatchGetModel	API 呼叫
AWS DeepLens	BatchGetProject	API 呼叫
AWS DeepLens	CreateDeviceCertificates	API 呼叫
AWS DeepRacer	AdminGetAccountConfig	API 呼叫
AWS DeepRacer	AdminListAssociatedUsers	API 呼叫
AWS DeepRacer	TestRewardFunction	API 呼叫
AWS DeepRacer	VerifyResourcesExistForTagris	API 呼叫
Amazon Detective	BatchGetGraphMemberDatasources	API 呼叫
Amazon Detective	BatchGetMembership Datasources	API 呼叫
Amazon Detective	SearchGraph	API 呼叫
Amazon DevOps 大師	SearchInsights	API 呼叫

服務	事件名稱	事件類型
Amazon DevOps 大師	SearchOrganizationInsights	API 呼叫
AWS Database Migration Service	BatchStartRecommendations	API 呼叫
AWS Database Migration Service	ModifyRecommendation	API 呼叫
AWS Database Migration Service	StartRecommendations	API 呼叫
AWS Database Migration Service	VerifyResourcesExistForTagris	API 呼叫
AWS Directory Service	VerifyTrust	API 呼叫
Amazon Elastic Compute Cloud	ConfirmProductInstance	API 呼叫
Amazon Elastic Compute Cloud	ReportInstanceStatus	API 呼叫
Amazon Elastic Container Registry	BatchCheckLayerAvailability	API 呼叫
Amazon Elastic Container Registry	BatchGetImage	API 呼叫
Amazon Elastic Container Registry	BatchGetImageReferrer	API 呼叫
Amazon Elastic Container Registry	BatchGetRepository ScanningConfiguration	API 呼叫
Amazon Elastic Container Registry	DryRunEvent	服務事件

服務	事件名稱	事件類型
Amazon Elastic Container Registry	PolicyExecutionEvent	服務事件
Amazon Elastic Container Registry Public	BatchCheckLayerAvailability	API 呼叫
Amazon Elastic Container Service	DiscoverPollEndpoint	API 呼叫
Amazon Elastic Container Service	FindSubfleetRoute	API 呼叫
Amazon Elastic Container Service	ValidateResources	API 呼叫
Amazon Elastic Container Service	VerifyTaskSetsExist	API 呼叫
Amazon Elastic Kubernetes Service	AccessKubernetesApi	API 呼叫
AWS Elastic Beanstalk	CheckDNSAvailability	API 呼叫
AWS Elastic Beanstalk	RequestEnvironmentInfo	API 呼叫
AWS Elastic Beanstalk	RetrieveEnvironmentInfo	API 呼叫
AWS Elastic Beanstalk	ValidateConfigurationSettings	API 呼叫
Amazon Elastic File System	NewClientConnection	服務事件
Amazon Elastic File System	UpdateClientConnection	服務事件
Amazon Elastic Transcoder	ReadJob	API 呼叫
Amazon Elastic Transcoder	ReadPipeline	API 呼叫
Amazon Elastic Transcoder	ReadPreset	API 呼叫

服務	事件名稱	事件類型
Amazon EventBridge	TestEventPattern	API 呼叫
Amazon EventBridge	TestScheduleExpression	API 呼叫
Amazon FinSpace API	BatchListCatalogNodesByDataset	API 呼叫
Amazon FinSpace API	BatchListNodesByDataset	API 呼叫
Amazon FinSpace API	BatchValidateAccess	API 呼叫
Amazon FinSpace API	CreateAuditRecordsQuery	API 呼叫
Amazon FinSpace API	SearchDatasets	API 呼叫
Amazon FinSpace API	SearchDatasetsV	API 呼叫
Amazon FinSpace API	ValidateIdToken	API 呼叫
AWS Firewall Manager	DisassociateAdminAccount	API 呼叫
Amazon Forecast	InvokeForecastEndpoint	API 呼叫
Amazon Forecast	QueryFeature	API 呼叫
Amazon Forecast	QueryForecast	API 呼叫
Amazon Forecast	QueryWhatIfForecast	API 呼叫
Amazon Forecast	VerifyResourcesExistForTagris	API 呼叫
Amazon Fraud Detector	BatchGetVariable	API 呼叫
Amazon Fraud Detector	VerifyResourcesExistForTagris	API 呼叫
FreeRTOS	VerifyEmailAddress	API 呼叫
Amazon GameLift	RequestUploadCredentials	API 呼叫

服務	事件名稱	事件類型
Amazon GameLift	ResolveAlias	API 呼叫
Amazon GameLift	SearchGameSessions	API 呼叫
Amazon GameLift	ValidateMatchmakingRuleSet	API 呼叫
Amazon GameSparks	ExportSnapshot	API 呼叫
Amazon Location Service	BatchGetDevicePosition	API 呼叫
Amazon Location Service	CalculateRoute	API 呼叫
Amazon Location Service	CalculateRouteMatrix	API 呼叫
Amazon Location Service	SearchPlaceIndexForPosition	API 呼叫
Amazon Location Service	SearchPlaceIndexForSuggestions	API 呼叫
Amazon Location Service	SearchPlaceIndexForText	API 呼叫
Amazon S3 Glacier	InitiateJob	API 呼叫
AWS Glue	BatchGetBlueprints	API 呼叫
AWS Glue	BatchGetColumnStatisticsForTable	API 呼叫
AWS Glue	BatchGetCrawlers	API 呼叫
AWS Glue	BatchGetCustomEntityTypes	API 呼叫
AWS Glue	BatchGetDataQualityResult	API 呼叫
AWS Glue	BatchGetDevEndpoints	API 呼叫
AWS Glue	BatchGetJobs	API 呼叫
AWS Glue	BatchGetML转换	API 呼叫

服務	事件名稱	事件類型
AWS Glue	BatchGetPartition	API 呼叫
AWS Glue	BatchGetTriggers	API 呼叫
AWS Glue	BatchGetWorkflows	API 呼叫
AWS Glue	QueryJobRuns	API 呼叫
AWS Glue	QueryJobRunsAggregated	API 呼叫
AWS Glue	QueryJobs	API 呼叫
AWS Glue	QuerySchemaVersion Metadata	API 呼叫
AWS Glue	SearchTables	API 呼叫
AWS HealthLake	ReadResource	API 呼叫
AWS HealthLake	SearchWithGet	API 呼叫
AWS HealthLake	SearchWithPost	API 呼叫
AWS Identity and Access Management	GenerateCredentialReport	API 呼叫
AWS Identity and Access Management	GenerateOrganizationsAccess Report	API 呼叫
AWS Identity and Access Management	GenerateServiceLas tAccessedDetails	API 呼叫
AWS Identity and Access Management	SimulateCustomPolicy	API 呼叫
AWS Identity and Access Management	SimulatePrincipalPolicy	API 呼叫
AWS 身分存放區	IsMemberInGroups	API 呼叫

服務	事件名稱	事件類型
AWS 身分存放區驗證	BatchGetSession	API 呼叫
Amazon Inspector Classic	PreviewAgents	API 呼叫
Amazon Inspector Classic	BatchGetAccountStatus	API 呼叫
Amazon Inspector Classic	BatchGetFreeTrialInfo	API 呼叫
Amazon Inspector Classic	BatchGetMember	API 呼叫
AWS Invoicing	ValidateDocumentDeliveryS3LocationInfo	API 呼叫
AWS IoT	SearchIndex	API 呼叫
AWS IoT	TestAuthorization	API 呼叫
AWS IoT	TestInvokeAuthorizer	API 呼叫
AWS IoT	ValidateSecurityProfileBehaviors	API 呼叫
AWS IoT Analytics	SampleChannelData	API 呼叫
AWS IoT SiteWise	GatewaysVerifyResourcesExistForTagrisInternal	API 呼叫
AWS IoT Things Graph	SearchEntities	API 呼叫
AWS IoT Things Graph	SearchFlowExecutions	API 呼叫
AWS IoT Things Graph	SearchFlowTemplates	API 呼叫
AWS IoT Things Graph	SearchSystemInstances	API 呼叫
AWS IoT Things Graph	SearchSystemTemplates	API 呼叫
AWS IoT Things Graph	SearchThings	API 呼叫
AWS IoT TwinMaker	ExecuteQuery	API 呼叫

服務	事件名稱	事件類型
AWS IoT Wireless	CreateNetworkAnalyzerConfiguration	API 呼叫
AWS IoT Wireless	DeleteNetworkAnalyzerConfiguration	API 呼叫
AWS IoT Wireless	DeregisterWirelessDevice	API 呼叫
Amazon Interactive Video Service	BatchGetChannel	API 呼叫
Amazon Interactive Video Service	BatchGetStreamKey	API 呼叫
Amazon Kendra	BatchGetDocumentStatus	API 呼叫
Amazon Kendra	Query	API 呼叫
Amazon Managed Service for Apache Flink	DiscoverInputSchema	API 呼叫
AWS Key Management Service	解密	API 呼叫
AWS Key Management Service	加密	API 呼叫
AWS Key Management Service	GenerateDataKey	API 呼叫
AWS Key Management Service	GenerateDataKeyPair	API 呼叫
AWS Key Management Service	GenerateDataKeyPairWithoutPlaintext	API 呼叫
AWS Key Management Service	GenerateDataKeyWithoutPlaintext	API 呼叫

服務	事件名稱	事件類型
AWS Key Management Service	GenerateMac	API 呼叫
AWS Key Management Service	GenerateRandom	API 呼叫
AWS Key Management Service	ReEncrypt	API 呼叫
AWS Key Management Service	符號	API 呼叫
AWS Key Management Service	確認	API 呼叫
AWS Key Management Service	VerifyMac	API 呼叫
AWS Lake Formation	SearchDatabasesByLF 標籤	API 呼叫
AWS Lake Formation	SearchTablesByLF 標籤	API 呼叫
AWS Lake Formation	StartQueryPlanning	API 呼叫
Amazon Lex	BatchCreateCustomVocabularyItem	API 呼叫
Amazon Lex	BatchDeleteCustomVocabularyItem	API 呼叫
Amazon Lex	BatchUpdateCustomVocabularyItem	API 呼叫
Amazon Lex	DeleteCustomVocabulary	API 呼叫
Amazon Lex	SearchAssociatedTranscripts	API 呼叫

服務	事件名稱	事件類型
Amazon Lightsail	創建 GUI SessionAccessDetails	API 呼叫
Amazon Lightsail	DownloadDefaultKeyPair	API 呼叫
Amazon Lightsail	IsVpcPeered	API 呼叫
Amazon CloudWatch 日誌	FilterLogEvents	API 呼叫
Amazon Macie	BatchGetCustomDataIdentifiers	API 呼叫
Amazon Macie	UpdateFindingsFilter	API 呼叫
AWS Elemental MediaConnect	ManagedDescribeFlow	API 呼叫
AWS Elemental MediaConnect	PrivateDescribeFlowMeta	API 呼叫
AWS Application Migration Service	OperationalDescribeJobLogItems	API 呼叫
AWS Application Migration Service	OperationalDescribeJobs	API 呼叫
AWS Application Migration Service	OperationalDescribeReplicationConfigurationTemplates	API 呼叫
AWS Application Migration Service	OperationalDescribeSourceServer	API 呼叫
AWS Application Migration Service	OperationalGetLaunchConfiguration	API 呼叫
AWS Application Migration Service	OperationalListSourceServers	API 呼叫

服務	事件名稱	事件類型
AWS Application Migration Service	VerifyClientRoleForMgn	API 呼叫
AWS HealthOmics	VerifyResourceExists	API 呼叫
AWS HealthOmics	VerifyResourcesExistForTagris	API 呼叫
Amazon Polly	SynthesizeLongSpeech	API 呼叫
Amazon Polly	SynthesizeSpeech	API 呼叫
Amazon Polly	SynthesizeSpeechGet	API 呼叫
AWS 提供託管專用網絡的服務	Ping	API 呼叫
AWS Proton	DeleteEnvironmentTemplateVersion	API 呼叫
AWS Proton	DeleteServiceTemplateVersion	API 呼叫
Amazon QLDB	ShowCatalog	API 呼叫
Amazon QuickSight	GenerateEmbedUrlForAnonymousUser	API 呼叫
Amazon QuickSight	GenerateEmbedUrlForRegisteredUser	API 呼叫
Amazon QuickSight	QueryDatabase	服務事件
Amazon QuickSight	SearchAnalyses	API 呼叫
Amazon QuickSight	SearchDashboards	API 呼叫
Amazon QuickSight	SearchDataSets	API 呼叫
Amazon QuickSight	SearchDataSources	API 呼叫

服務	事件名稱	事件類型
Amazon QuickSight	SearchFolders	API 呼叫
Amazon QuickSight	SearchGroups	API 呼叫
Amazon QuickSight	SearchUsers	API 呼叫
Amazon Relational Database Service	DownloadComplete資料庫 LogFile	API 呼叫
Amazon Relational Database Service	下載資料庫 LogFilePortion	API 呼叫
Amazon Rekognition	CompareFaces	API 呼叫
Amazon Rekognition	DetectCustomLabels	API 呼叫
Amazon Rekognition	DetectFaces	API 呼叫
Amazon Rekognition	DetectLabels	API 呼叫
Amazon Rekognition	DetectModerationLabels	API 呼叫
Amazon Rekognition	DetectProtectiveEquipment	API 呼叫
Amazon Rekognition	DetectText	API 呼叫
Amazon Rekognition	RecognizeCelebrities	API 呼叫
Amazon Rekognition	SearchFaces	API 呼叫
Amazon Rekognition	SearchFacesByImage	API 呼叫
Amazon Rekognition	SearchUsers	API 呼叫
Amazon Rekognition	SearchUsersByImage	API 呼叫
AWS 資源總管	BatchGetView	API 呼叫
AWS 資源總管	搜尋	API 呼叫

服務	事件名稱	事件類型
AWS Resource Groups	SearchResources	API 呼叫
AWS Resource Groups	ValidateResourceSharing	API 呼叫
AWS RoboMaker	BatchDescribeSimulationJob	API 呼叫
Amazon Route 53	TestDNSAnswer	API 呼叫
Amazon Route 53 網域	checkAvailabilities	API 呼叫
Amazon Route 53 網域	CheckDomainAvailability	API 呼叫
Amazon Route 53 網域	checkDomainTransferability	API 呼叫
Amazon Route 53 網域	CheckDomainTransferability	API 呼叫
Amazon Route 53 網域	isEmailReachable	API 呼叫
Amazon Route 53 網域	searchDomains	API 呼叫
Amazon Route 53 網域	sendVerificationMessage	API 呼叫
Amazon Route 53 網域	ViewBilling	API 呼叫
Amazon Route 53 網域	viewBilling	API 呼叫
Amazon CloudWatch 朗姆	BatchGetRumMetricDefinitions	API 呼叫
Amazon Simple Storage Service	回應	API 呼叫
Amazon Simple Storage Service	GenerateInventory	服務事件
Amazon SageMaker	BatchDescribeModelPackage	API 呼叫
Amazon SageMaker	DeleteModelCard	API 呼叫
Amazon SageMaker	QueryLineage	API 呼叫

服務	事件名稱	事件類型
Amazon SageMaker	RenderUiTemplate	API 呼叫
Amazon SageMaker	搜尋	API 呼叫
Amazon EventBridge 模式	ExportSchema	API 呼叫
Amazon EventBridge 模式	SearchSchemas	API 呼叫
Amazon SimpleDB	DomainMetadata	API 呼叫
AWS Secrets Manager	ValidateResourcePolicy	API 呼叫
AWS Service Catalog	ScanProvisionedProducts	API 呼叫
AWS Service Catalog	SearchProducts	API 呼叫
AWS Service Catalog	SearchProductsAsAdmin	API 呼叫
AWS Service Catalog	SearchProvisionedProducts	API 呼叫
Amazon SES	BatchGetMetricData	API 呼叫
Amazon SES	TestRenderEmailTemplate	API 呼叫
Amazon SES	TestRenderTemplate	API 呼叫
Amazon Simple Notification Service	CheckIfPhoneNumber IsOptedOut	API 呼叫
AWS SQL Workbench	BatchGetNotebookCell	API 呼叫
AWS SQL Workbench	ExportNotebook	API 呼叫
Amazon EC2 Systems Manager	ExecuteApi	API 呼叫
AWS Systems Manager Incident Manager	DeleteContactChannel	API 呼叫
AWS IAM Identity Center	IsMemberInGroup	API 呼叫

服務	事件名稱	事件類型
AWS IAM Identity Center	SearchGroups	API 呼叫
AWS IAM Identity Center	SearchUsers	API 呼叫
AWS STS	AssumeRole	API 呼叫
AWS STS	AssumeRoleWith薩姆爾	API 呼叫
AWS STS	AssumeRoleWithWebIdentity	API 呼叫
AWS STS	DecodeAuthorizationMessage	API 呼叫
AWS 稅金設定	BatchGetTaxExemptions	API 呼叫
AWS WAFV2	CheckCapacity	API 呼叫
AWS WAFV2	GenerateMobileSdkR eleaseUrl	API 呼叫
AWS Well-Architected Tool	ExportLens	API 呼叫
AWS Well-Architected Tool	TagResource	API 呼叫
AWS Well-Architected Tool	UntagResource	API 呼叫
AWS Well-Architected Tool	UpdateGlobalSettings	API 呼叫
Amazon Connect Wisdom	QueryAssistant	API 呼叫
Amazon Connect Wisdom	SearchContent	API 呼叫
Amazon Connect Wisdom	SearchSessions	API 呼叫
Amazon WorkDocs	AbortDocumentVersionUpload	API 呼叫
Amazon WorkDocs	AddUsersToGroup	API 呼叫
Amazon WorkDocs	BatchGetUsers	API 呼叫
Amazon WorkDocs	CheckAlias	API 呼叫

服務	事件名稱	事件類型
Amazon WorkDocs	CompleteDocumentVersionUpload	API 呼叫
Amazon WorkDocs	CreateAnnotation	API 呼叫
Amazon WorkDocs	CreateComment	API 呼叫
Amazon WorkDocs	CreateFeedbackRequest	API 呼叫
Amazon WorkDocs	CreateFolder	API 呼叫
Amazon WorkDocs	CreateGroup	API 呼叫
Amazon WorkDocs	CreateShare	API 呼叫
Amazon WorkDocs	CreateUser	API 呼叫
Amazon WorkDocs	DeleteAnnotation	API 呼叫
Amazon WorkDocs	DeleteComment	API 呼叫
Amazon WorkDocs	DeleteDocument	API 呼叫
Amazon WorkDocs	DeleteFeedbackRequest	API 呼叫
Amazon WorkDocs	DeleteFolder	API 呼叫
Amazon WorkDocs	DeleteFolderContents	API 呼叫
Amazon WorkDocs	DeleteGroup	API 呼叫
Amazon WorkDocs	DeleteOrganizationShare	API 呼叫
Amazon WorkDocs	DeleteUser	API 呼叫
Amazon WorkDocs	DownloadDocumentVersion	API 呼叫
Amazon WorkDocs	DownloadDocumentVersionUnderlays	API 呼叫

服務	事件名稱	事件類型
Amazon WorkDocs	InitiateDocumentVersionUpload	API 呼叫
Amazon WorkDocs	LogoutUser	API 呼叫
Amazon WorkDocs	PaginatedOrganizationActivity	API 呼叫
Amazon WorkDocs	PublishAnnotations	API 呼叫
Amazon WorkDocs	PublishComments	API 呼叫
Amazon WorkDocs	RestoreDocument	API 呼叫
Amazon WorkDocs	RestoreFolder	API 呼叫
Amazon WorkDocs	SearchGroups	API 呼叫
Amazon WorkDocs	SearchOrganizationUsers	API 呼叫
Amazon WorkDocs	TransferUserResources	API 呼叫
Amazon WorkDocs	UpdateAnnotation	API 呼叫
Amazon WorkDocs	UpdateComment	API 呼叫
Amazon WorkDocs	UpdateDocument	API 呼叫
Amazon WorkDocs	UpdateDocumentVersion	API 呼叫
Amazon WorkDocs	UpdateFolder	API 呼叫
Amazon WorkDocs	UpdateGroup	API 呼叫
Amazon WorkDocs	UpdateOrganization	API 呼叫
Amazon WorkDocs	UpdateUser	API 呼叫
Amazon WorkMail	AssumeImpersonationRole	API 呼叫
Amazon WorkMail	QueryDnsRecords	API 呼叫

服務	事件名稱	事件類型
Amazon WorkMail	SearchMembers	API 呼叫
Amazon WorkMail	TestAvailabilityConfiguration	API 呼叫
Amazon WorkMail	TestInboundMailFlowRules	API 呼叫
Amazon WorkMail	TestOutboundMailFlowRules	API 呼叫

EventBridge 事件詳細參考

EventBridge 本身發出以下事件。與任何其他 AWS 服務一樣，這些事件會自動傳送至預設事件匯流排。

如需包含在所有事件中之中繼資料欄位的定義，請參閱the section called “事件結構參考”。

主題

- 已排程事件
- 架構已建立
- 架構版本已建立

已排程事件

以下是Scheduled Event事件的詳細信息字段。

source和detail-type欄位會包含在內，因為它們包含 EventBridge 事件的特定值。如需包含在所有事件中的其他中繼資料欄位的定義，請參閱the section called “事件結構參考”。

```
{
  . . .,
  "detail-type": "Scheduled Event",
  "source": "aws.events",
  . . .,
  "detail": {}
}
```

`detail-type`

識別事件的類型。

對於此事件，此值為`Scheduled Event`。

必要：是

`source`

識別產生事件的服務。對於 EventBridge 事件，此值為`aws.events`。

必要：是

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

必要：是

此物件中沒有`Scheduled Event`事件的必填欄位。

Example 排程事件事件範例

```
{
  "version": "0",
  "id": "89d1a02d-5ec7-412e-82f5-13505f849b41",
  "detail-type": "Scheduled Event",
  "source": "aws.events",
  "account": "123456789012",
  "time": "2016-12-30T18:44:49Z",
  "region": "us-east-1",
  "resources": ["arn:aws:events:us-east-1:123456789012:rule/SampleRule"],
  "detail": {}
}
```

架構已建立

以下是`Schema Created`事件的詳細信息字段。

建立結構描述時， EventBridge 會同時傳送`Schema Created`和`Schema Version Created`事件。

`source`和`detail-type`欄位會包含在內，因為它們包含 EventBridge 事件的特定值。如需包含在所有事件中的其他中繼資料欄位的定義，請參閱the section called “事件結構參考”。

```
{
  . . .,
  "detail-type": "Schema Created",
  "source": "aws.schemas",
  . . .,
  "detail": {
    "SchemaName" : "String",
    "SchemaType" : "String",
    "RegistryName" : "String",
    "CreationDate" : "DateTime",
    "Version" : "Number"
  }
}
```

`detail-type`

識別事件的類型。

對於此事件，此值為`Schema Created`。

必要：是

`source`

識別產生事件的服務。對於 EventBridge 事件，此值為`aws.schemas`。

必要：是

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

必要：是

對於此事件，此資料包括：

`SchemaName`

結構描述的名稱。

必要：是

`SchemaType`

結構描述的類型。

有效值：`OpenApi3` | `JSONSchemaDraft4`

必要：是

`RegistryName`

包含結構描述的登錄檔的名稱。

必要：是

`CreationDate`

建立結構描述的日期。

必要：是

`Version`

此結構描述的版本。

對於`Schema Created`事件，這個值將永遠是`1`。

必要：是

Example 範例架構已建立事件

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Schema Created",
  "source": "aws.schemas",
  "account": "123456789012",
  "time": "2019-05-31T21:49:54Z",
  "region": "us-east-1",
  "resources": ["arn:aws:schemas:us-east-1::schema/myRegistry/mySchema"],
  "detail": {
    "SchemaName": "mySchema",
    "SchemaType": "OpenApi3",
    "RegistryName": "myRegistry",
    "CreationDate": "2019-11-29T20:08:55Z",
    "Version": "1"
  }
}
```

架構版本已建立

以下是`Schema Version Created`事件的詳細信息字段。

建立結構描述時，EventBridge 會同時傳送Schema Created和Schema Version Created事件。

source和detail-type欄位會包含在內，因為它們包含 EventBridge 事件的特定值。如需包含在所有事件中的其他中繼資料欄位的定義，請參閱the section called “事件結構參考”。

```
{
  . . .,
  "detail-type": "Schema Version Created",
  "source": "aws.schemas",
  . . .,
  "detail": {
    "SchemaName" : "String",
    "SchemaType" : "String",
    "RegistryName" : "String",
    "CreationDate" : "DateTime",
    "Version" : "Number"
  }
}
```

detail-type

識別事件的類型。

對於此事件，此值為Schema Version Created。

必要：是

source

識別產生事件的服務。對於 EventBridge 事件，此值為aws.schemas。

必要：是

detail

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

必要：是

對於此事件，此資料包括：

SchemaName

結構描述的名稱。

必要：是

SchemaType

結構描述的類型。

有效值：OpenApi3 | JSONSchemaDraft4

必要：是

RegistryName

包含結構描述的登錄檔的名稱。

必要：是

CreationDate

建立結構描述版本的日期。

必要：是

Version

此結構描述的版本。

必要：是

Example 範例架構版本已建立事件

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Schema Version Created",
  "source": "aws.schemas",
  "account": "123456789012",
  "time": "2019-05-31T21:49:54Z",
  "region": "us-east-1",
  "resources": ["arn:aws:schemas:us-east-1::schema/myRegistry/mySchema"],
  "detail": {
    "SchemaName": "mySchema",
    "SchemaType": "OpenApi3",
    "RegistryName": "myRegistry",
    "CreationDate": "2019-11-29T20:08:55Z",
    "Version": "5"
  }
}
```

接收來自 SaaS 合作夥伴與 Amazon 的事件 EventBridge

若要從 SaaS 合作夥伴應用程式和服務接收事件，您必須擁有合作夥伴提供給您的合作夥伴事件來源。然後，您可以建立合作夥伴事件匯流排，並將其與對應的合作夥伴事件來源進行比對。

以下影片涵蓋與以下項目的 SaaS 整合 EventBridge：軟體即服務 (SaaS) 合作夥伴

主題

- 支援 SaaS 合作夥伴整合
- 配置 Amazon EventBridge 以接收來自 SaaS 集成的事件
- 建立符合 SaaS 合作夥伴事件的規則
- 使用 AWS Lambda 函數 URL 接收事件
- 接收來自 Salesforce 的事件

支援 SaaS 合作夥伴整合

EventBridge 支援下列 SaaS 合作夥伴整合：

- Adobe
- Auth0
- Blitline
- BUIDLHub
- Buildkite
- CleverTap
- Datadog
- Epsagon
- Freshworks
- Genesys
- GS2
- Karte
- Kloudless
- Mackerel
- MongoDB

- New Relic
- OneLogin
- Opsgenie
- PagerDuty
- Payshield
- SaaSus Platform
- SailPoint
- Saviynt
- Segment
- Shopify
- SignalFx
- Site24x7
- Stax
- Stripe
- SugarCRM
- SugarCRM
- Symantec
- Thundra
- TriggerMesh
- Whispir
- Zendesk
- Amazon 賣家合作夥伴 API

以下區域提供合作夥伴事件來源。

Code	名稱
us-east-1	美國東部 (維吉尼亞北部)
us-east-2	美國東部 (俄亥俄)
us-west-1	美國西部 (加利佛尼亞北部)

Code	名稱
us-west-2	美國西部 (奧勒岡)
ca-central-1	加拿大 (中部)
eu-central-1	歐洲 (法蘭克福)
eu-central-2	歐洲 (蘇黎世)
eu-west-1	歐洲 (愛爾蘭)
eu-west-2	歐洲 (倫敦)
eu-west-3	歐洲 (巴黎)
eu-north-1	歐洲 (斯德哥爾摩)
eu-south-1	歐洲 (米蘭)
eu-south-2	歐洲 (西班牙)
af-south-1	非洲 (開普敦)
ap-south-1	亞太區域 (孟買)
ap-south-2	亞太區域 (海德拉巴)
ap-east-1	亞太區域 (香港)
ap-northeast-1	亞太區域 (東京)
ap-northeast-2	亞太區域 (首爾)
ap-northeast-3	亞太區域 (大阪)
ap-southeast-1	亞太區域 (新加坡)
ap-southeast-2	亞太區域 (悉尼)
ap-southeast-3	亞太區域 (雅加達)

Code	名稱
ap-southeast-4	亞太區域 (墨爾本)
cn-north-1	中國 (北京)
cn-northwest-1	中國 (寧夏)
me-central-1	中東 (阿拉伯聯合大公國)
me-south-1	Middle East (Bahrain)
sa-east-1	南美洲 (聖保羅)
il-central-1	以色列 (特拉維夫)

配置 Amazon EventBridge 以接收來自 SaaS 集成的事件

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中, 選擇合作夥伴事件來源。
3. 尋找您想要的合作夥伴，然後為該合作夥伴選擇設定。
4. 若要將您的帳戶 ID 複製到剪貼簿，選擇複製。
5. 在導覽窗格中， 選擇合作夥伴事件來源。
6. 前往合作夥伴的網站，並依照指示使用您的账户 ID 建立合作夥伴事件來源。您建立的事件來源將僅供帳戶使用。
7. 返回 EventBridge 主控台，然後在導覽窗格中選擇 [合作夥伴事件來源]。
8. 選取合作夥伴事件來源旁邊的按鈕，然後選擇與事件匯流排建立關聯。

   該事件來源的狀態會從 Pending 變更為 Active，而且事件匯流排的名稱會更新，以符合合作夥伴事件來源名稱。您現在可以開始從該合作夥伴事件來源建立符合事件的規則。如需詳細資訊，請參閱 建立符合 SaaS 合作夥伴事件的規則。

   > **Note**
   >
   > 由合作夥伴發佈至合作夥伴事件來源，但尚未與事件匯流排相關聯的任何事件都會立即遭到捨棄。這些事件將不會在休息時持續存在 EventBridge。

建立符合 SaaS 合作夥伴事件的規則

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇規則。
3. 選擇建立規則。
4. 輸入規則的名稱和描述。

   在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
5. 針對事件匯流排，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 AWS 預設事件匯流排。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對規則類型，選擇具有事件模式的規則。
7. 選擇下一步。
8. 在事件來源中，選擇其他。
9. (選用)針對範例事件，請選擇事件類型。
10. 在事件模式中，輸入 JSON 事件模式。
11. 選擇下一步。
12. 在目標類型欄位中，選擇 AWS 服務。
13. 針對 [選取目標]，選擇當 EventBridge 偵測到符合事件模式的事件時，要傳送資訊的 AWS 服務。
14. 顯示的欄位會根據您選擇的服務而有所不同。請視需要輸入此目標類型的特定資訊。
15. 對於許多目標類型， EventBridge 需要將事件傳送至目標的權限。在這些情況下， EventBridge 可以建立執行規則所需的 IAM 角色。執行以下任意一項：

    - 若要自動建立 IAM 角色，請選擇為此特定資源建立新角色。
    - 若要使用您早前建立的 IAM 角色，請選擇使用現有角色並從下拉式清單中選取現有角色。
16. (選用) 針對其他設定，請執行下列動作：

    a. 針對 Maximum age of event (事件的最長存留期)，輸入介於一分鐘 (00:01) 到 24 小時 (24:00) 之間的某個值。

    b. 針對重試嘗試，輸入介於 0 到 185 之間的某個數。

    c. 對於無效字母佇列，請選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。 EventBridge 如果符合此規則的事件未成功傳遞至目標，則會將符合此規則的事件傳送至無效字母佇列。執行以下任意一項：

       - 選擇無，即不使用無效字母佇列。

- 選擇選取當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列，然後從下拉式清單中選取要使用的佇列。
- 選擇選取其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列，然後輸入要使用的佇列 ARN。您必須將以資源為基礎的政策附加至佇列，以授與傳送訊息給該佇列的 EventBridge 權限。如需詳細資訊，請參閱 將許可授予無效字母佇列。

17. (選用) 選擇新增其他目標，為此規則新增另一個目標。
18. 選擇下一步。
19. (選用) 為規則輸入一或多個標籤。如需詳細資訊，請參閱 Amazon EventBridge 標籤。
20. 選擇下一步。
21. 檢閱規則的詳細資訊，然後選擇建立規則。

使用 AWS Lambda 函數 URL 接收事件

> **Note**
>
> 為了讓我們的合作夥伴能夠存取輸入 Webhook，我們在您的 AWS 帳戶中建立一個 Open Lambda，藉由驗證第三方合作夥伴傳送的驗證簽章，在 Lambda 應用程式層級保護。請與您的安全團隊一起檢閱此組態。如需詳細資訊，請參閱 Lambda 函數 URL 的安全性和驗證模型。

您的 Amazon EventBridge 事件匯流排可以使用 AWS CloudFormation 範本建立的AWS Lambda 函數 URL，從支援的 SaaS 供應商接收事件。在函數 URL，事件資料會傳送至 Lambda 函數。然後，函數會將此資料轉換成可由擷取 EventBridge 並傳送至事件匯流排以進行處理的事件。事件位於事件匯流排上之後，您可以使用規則來篩選事件、套用任何已設定的輸入轉換，然後將其路由至正確的目標。

> **Note**
>
> 建立 Lambda 函數 URL 網址會增加您的每月成本。如需詳細資訊，請參閱 AWS Lambda 定價。

若要設定連線 EventBridge，請先選取要設定連線的 SaaS 提供者。然後，您提供使用該提供者建立的簽署密碼，然後選取要傳送 EventBridge 事件的事件匯流排。最後，您可以使用 AWS CloudFormation 範本並建立完成連線所需的資源。

下列 SaaS 提供者目前可與使用 Lambda 函數 URL 搭配 EventBridge 使用：

- GitHub
- Stripe
- Twilio

主題

- 設定與 Twilio 的連線
- 更新 Webhook 機密或身分驗證權杖
- 更新 Lambda 函數
- 可用事件類型
- 配額、錯誤碼和重試傳送

設定與 GitHub 的連線

步驟 1：建立 AWS CloudFormation 堆疊

首先，使用 Amazon EventBridge 控制台創建一個 CloudFormation 堆棧：

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 從導覽窗格選擇快速入門。
3. 在使用 Lambda fURLs 的傳入 Webhook下，選擇開始使用。
4. 在 GitHub 下，選擇設定。
5. 在步驟 1：選取事件匯流排下，從下拉式清單中選取事件匯流排。此事件匯流排會從您提供給 GitHub 的 Lambda 函數 URL 接收資料。您也可以選取新事件匯流排來建立事件匯流排。
6. 在「步驟 2：設定方式」下 CloudFormation，選擇「新增GitHub網路掛接」。
7. 選取我確認我建立的傳入 Webhook 將可公開存取。然後選擇確認。
8. 輸入堆疊名稱。
9. 在參數下，確認已列出正確的事件匯流排，然後為 GitHubWebhookSecret 確定安全字符。有關建立安全字符的更多信息，請參閱 GitHub 文件中的設定私密字符。
10. 在功能和轉換下，選取下列每一項：
    - 我承認 AWS CloudFormation 可能會建立 IAM 資源。
    - 我承認 AWS CloudFormation 可能會使用自定義名稱創建 IAM 資源。
    - 我確認 AWS CloudFormation 可能需要以下功能：**`CAPABILITY_AUTO_EXPAND`**
11. 選擇建立堆疊。

步驟 2：建立 GitHub Webhook

接下來，建立 GitHub 上的 Webhook。您需要使用在步驟 2 中建立的安全權杖和 Lambda 函數 URL完成該步驟。如需詳細資訊，請參閱文件 GitHub 中的建立 Webhook。

設定與 Stripe 的連線

步驟 1：建立 Stripe 端點

若要設定 EventBridge 和 Stripe 之間的連線，請先建立 Stripe 端點並記下端點密碼。在步驟 2 中設定堆疊需使用該端點機密。如需詳細資訊，請參閱 Stripe 文件中的互動式 Webhook 端點建置器。

> **Note**
>
> 您需透過虛擬 URL 使用 Stripe 設定端點。例如 `www.example.com`。

步驟 2：建立 AWS CloudFormation 堆疊

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇快速入門。
3. 在使用 Lambda fURLs 的傳入 Webhook下，選擇開始使用。
4. 在 Stripe 下，選擇設定。
5. 在步驟 1：選取和啟用事件匯流排下，從下拉式清單中選取事件匯流排。此事件匯流排會從您提供給 Stripe 的 Lambda 函數 URL 接收資料。您也可以選取新事件匯流排來建立事件匯流排。
6. 在「步驟 2：設定方式」下 CloudFormation，選擇「新增Stripe網路掛接」。
7. 選取我確認我建立的傳入 Webhook 將可公開存取。然後選擇確認。
8. 輸入堆疊名稱。
9. 在參數設定中，確認已列出正確的事件匯流排，然後輸入您在步驟 1 中建立的 StripeWebhookSecret。
10. 在功能和轉換下，選取下列每一項：
    - 我承認 AWS CloudFormation 可能會建立 IAM 資源。
    - 我承認 AWS CloudFormation 可能會使用自定義名稱創建 IAM 資源。
    - 我確認 AWS CloudFormation 可能需要以下功能：**`CAPABILITY_AUTO_EXPAND`**
11. 選擇建立堆疊。

步驟 3：更新 Stripe 端點

現在您已經建立了 Lambda 函數 URL，請更新 Stripe 端點傳送事件至 Lambda 函數 URL。

設定與 Twilio 的連線

步驟 1：尋找您的 Twilio 身分驗證權杖

要在Twilio和之間設置連接 EventBridge，請首先為您的Twilio帳戶設置Twilio與身份驗證令牌或秘密的連接。如需詳細資訊，請參閱 Twilio 文件中的身分驗證權杖和變更方法。

步驟 2：建立 AWS CloudFormation 堆疊

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇快速入門。
3. 在使用 Lambda fURLs 的傳入 Webhook下，選擇開始使用。
4. 在 Twilio 下，選擇設定。
5. 在步驟 1：選取和啟用事件匯流排下，從下拉式清單中選取事件匯流排。此事件匯流排會從提供給 Twilio 的 Lambda 函數 URL 接收資料。您也可以選取新事件匯流排來建立事件匯流排。
6. 在「步驟 2：設定方式」下 CloudFormation，選擇「新增Twilio網路掛接」。
7. 選取我確認我建立的傳入 Webhook 將可公開存取。然後選擇確認。
8. 輸入堆疊名稱。
9. 在參數設定中，確認已列出正確的事件匯流排，然後輸入您在步驟 1 中建立的 TwilioWebhookSecret。
10. 在功能和轉換下，選取下列每一項：
    - 我承認 AWS CloudFormation 可能會建立 IAM 資源。
    - 我承認 AWS CloudFormation 可能會使用自定義名稱創建 IAM 資源。
    - 我確認 AWS CloudFormation 可能需要以下功能：能力 _ 自動展開
11. 選擇建立堆疊。

步驟 3：建立 Twilio Webhook

在設定 Lambda 函數 URL 之後，您需要將其提供給 Twilio，以便能夠傳送事件資料。如需詳細資訊，請參閱 Twilio 文件中的使用 Twilio 設定公用 URL。

更新 Webhook 機密或身分驗證權杖

更新 GitHub 機密

> **Note**
>
> GitHub 不支援同時擁有兩個機密。當 AWS CloudFormation 堆疊中的GitHub密碼和密碼不同步時，您可能會遇到資源停機時間。 GitHub密碼不同步時傳送的訊息將會失敗，因為簽章不正確。請等到GitHub和 CloudFormation 密碼同步，然後再試一次。

1. 建立新 GitHub 機密。如需詳細資訊，請參閱 GitHub 文件中的加密機密。
2. 開啟主 AWS CloudFormation 控台，網址為 https://console.aws.amazon.com/cloudformation。
3. 從導覽窗格選擇堆疊。
4. 選擇用於 Webhook 的堆疊，該 Webhook 包含您準備更新的機密。
5. 選擇更新。
6. 確保已選取使用目前範本，然後選擇下一步。
7. 在下 GitHubWebhookSecret，清除使用現有值，輸入您在步驟 1 中建立的新GitHub密碼，然後選擇下一步。
8. 選擇下一步。
9. 請選擇更新堆疊。

機密的傳播可能需要長達一小時的時間。若要減少停機時間，可更新 Lambda 執行內容。

更新 Stripe 機密

1. 在 Stripe 儀表板的 Webhook 區段中，選取滾動機密，並將停機時間延遲至少兩 (2) 小時。如需詳細資訊，請參閱 Stripe 文件中的滾動端點機密。
2. 開啟主 AWS CloudFormation 控台，網址為 https://console.aws.amazon.com/cloudformation。
3. 從導覽窗格選擇堆疊。
4. 選擇用於 Webhook 的堆疊，該 Webhook 包含您準備更新的機密。
5. 選擇更新。
6. 確保已選取使用目前範本，然後選擇下一步。

7. 在下 StripeWebhookSecret，清除使用現有值，輸入您在步驟 1 中建立的新Stripe密碼，然後選擇下一步。
8. 選擇下一步。
9. 請選擇更新堆疊。

Stripe 將在輪換期間同時傳送舊簽名和新簽名。

更新 Twilio 機密

> **Note**
>
> Twilio 不支援同時擁有兩個機密。當 AWS CloudFormation 堆疊中的Twilio密碼和密碼不同步時，您可能會遇到資源停機時間。 Twilio密碼不同步時傳送的訊息會因為簽章不正確而失敗。請等到Twilio和 CloudFormation 密碼同步，然後再試一次。

1. 建立新 Twilio 機密。如需詳細資訊，請參閱 Twilio 文件中的身分驗證權杖和變更方法。
2. 開啟主 AWS CloudFormation 控台，網址為 https://console.aws.amazon.com/cloudformation。
3. 從導覽窗格選擇堆疊。
4. 選擇用於 Webhook 的堆疊，該 Webhook 包含您準備更新的機密。
5. 選擇更新。
6. 確保已選取使用目前範本，然後選擇下一步。
7. 在下 TwilioWebhookSecret，清除使用現有值，輸入您在步驟 1 中建立的新Twilio密碼，然後選擇下一步。
8. 選擇下一步。
9. 請選擇更新堆疊。

機密的傳播可能需要長達一小時的時間。若要減少停機時間，可更新 Lambda 執行內容。

更新 Lambda 函數

由 CloudFormation 堆疊所建立的 Lambda 函式會建立基本的網路掛接。如果您想要針對特定使用案例 (例如自訂記錄) 自訂 Lambda 函數，請使用 CloudFormation 主控台存取函數，然後使用 Lambda 主控台更新 Lambda 函數程式碼。

存取 Lambda 函數

1. 開啟主 AWS CloudFormation 控台，網址為 https://console.aws.amazon.com/cloudformation。
2. 從導覽窗格選擇堆疊。
3. 選擇用於 Webhook 的堆疊，該 Webhook 包含您準備更新的 Lambda 函數。
4. 選擇資源標籤。
5. 若要在 Lambda 主控台中開啟 Lambda 函數，請在實體 ID 下選擇 Lambda 函數的 ID。

現在您已存取 Lambda 函數，請使用 Lambda 主控台更新函數程式碼。

更新 Lambda 函數程式碼

1. 在動作下，選擇匯出函數。
2. 選擇下載部署套件並將檔案儲存至電腦。
3. 解壓縮部署套件.zip 檔案，更新 `app.py` 檔案，然後壓縮更新的部署套件，確保原始 .zip 檔案中的所有檔案均包含在內。
4. 在 Lambda 主控台中，選擇程式碼標籤。
5. 在程式碼來源下，選擇上傳自。
6. 選擇 .zip 檔案，然後選擇 上傳。
   - 在檔案選擇器中，選取已更新檔案，選擇開啟，然後選擇儲存。
7. 在動作下，選擇發佈新版本。

可用事件類型

事件匯流排目前支援下列 CloudFormation 事件類型：

- GitHub— 支援所有事件類型。
- Stripe：支援所有事件類型。
- Twilio：支援事件後 Webhook。

配額、錯誤碼和重試傳送

配額

對 webhook 的傳入請求數量由基礎 AWS 服務限制。下表包含相關配額。

服務	配額
AWS Lambda	預設值：10 個並行執行 如需有關配額的詳細資訊，包括請求增加配額，請參閱 Lambda 配額。
AWS Secrets Manager	預設值：5,000 (每秒請求數) 如需有關配額的詳細資訊，包括請求增加配額，請參閱 AWS Secrets Manager 配額。 **Note** 使用 AWS Secrets Manager Python 緩存客戶端，以使每秒請求數量最小。
Amazon EventBridge	動作的最大輸入大小為 PutEvents 256KB。 EventBridge 強制執行以區域為基礎的費率配額。如需詳細資訊，請參閱 ???。

錯誤代碼

發生錯誤時，每個 AWS 服務都會傳回特定的錯誤碼。下表包含相關錯誤代碼。

服務	錯誤代碼	描述
AWS Lambda	429「」TooManyRequestsException	超過並行執行配額。
AWS Secrets Manager	500「內部伺服器錯誤」	超過每秒配額的請求數。
Amazon EventBridge	500「內部伺服器錯誤」	超過基於區域的費率配額。

重新傳遞事件

發生錯誤時，您可以重新傳遞受影響的事件。每個 SaaS 提供者都有不同的重新傳遞程序。

GitHub

使用 GitHub Webhook API 檢查任何 Webhook 呼叫的傳遞狀態，並視需要重新傳遞事件。如需詳細資訊，請參閱下列 GitHub 文件：

- 組織：重新傳遞組織 Webhook
- 儲存庫：重新傳遞儲存庫 Webhook
- 應用程式：重新傳遞應用程式 Webhook

Stripe

Stripe 嘗試以指數退避方式交付您的 Webhook 長達三天。如需詳細資訊，請參閱下列 Stripe 文件：

- 傳遞嘗試和重試
- 錯誤處理

Twilio

Twilio 使用者可以使用連線覆寫來自訂事件重試選項。如需詳細資訊，請參閱 Twilio 文件中的 Webhook (HTTP 回呼)：連線覆寫。

接收來自 Salesforce 的事件

您可以使用 Amazon EventBridge 通過以下方式接收事件：Salesforce

- 透過使用Salesforce's事件匯流排中繼功能，直接在 EventBridge 合作夥伴活動匯流排上接收活動。
- 透過在 Amazon 中設定 AppFlow用Salesforce作資料來源的流程。 AppFlow 然後，Amazon 使用合作夥伴事件總線將事件發送Salesforce到 EventBridge 。

您可以使用 API 目的地傳送事件資訊至 Salesforce。一旦事件傳送至 Salesforce，流程或 Apex 觸發程序即可處理該事件。如需設定 Salesforce API 目的地的詳細資訊，請參閱 ???。

主題

- 使用事件匯流排中繼接收來自 Salesforce 的事件
- 從Salesforce使用 Amazon 接收事件 AppFlow

使用事件匯流排中繼接收來自 Salesforce 的事件

步驟 1：設定Salesforce事件匯流排轉送和 EventBridge 合作夥伴事件來源

當您在上建立事件轉送組態時Salesforce，Salesforce會建立處於擱置狀態 EventBridge 的夥伴事件來源。

若要設定 Salesforce 事件匯流排中繼，

1. 設定 REST API 工具
2. (選用) 定義平台事件
3. 為自訂平台事件建立頻道
4. 建立頻道會員以建立自訂平台事件的關聯
5. 建立命名憑證
6. 建立事件轉送組態

步驟 2：在 EventBridge 主控台中啟用Salesforce合作夥伴事件來源並啟動事件轉送

1. 在主控台中開啟 [合作夥伴事件來源] 頁 EventBridge 面。
2. 選取您在步驟 1 中建立的 Salesforce 合作夥伴事件來源。

3. 選擇與事件匯流排建立關聯。
4. 驗證合作夥伴事件匯流排的名稱。
5. 選擇關聯。
6. 啟動事件轉送

現在，您已設定並啟動事件匯流排轉送並設定合作夥伴事件來源，您可以建立對事件做出反應的 EventBridge 規則，以篩選資料並將資料傳送至目標。

從Salesforce使用 Amazon 接收事件 AppFlow

Amazon 將事件 AppFlow 封裝Salesforce在 EventBridge 事件信封中。下列範例顯示 EventBridge 合作夥伴Salesforce事件匯流排所接收的事件。

```
{
    "version": "0",
    "id": "5c42b99e-e005-43b3-c744-07990c50d2cc",
    "detail-type": "AccountChangeEvent",
    "source": "aws.partner/appflow.test/salesforce.com/364228160620/CustomSF-Source-
Final",
    "account": "000000000",
    "time": "2020-08-20T18:25:51Z",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "ChangeEventHeader": {
            "commitNumber": 248197218874,
            "commitUser": "0056g000003XW7AAAW",
            "sequenceNumber": 1,
            "entityName": "Account",
            "changeType": "UPDATE",
            "changedFields": [
                "LastModifiedDate",
                "Region__c"
            ],
            "changeOrigin": "com/salesforce/api/soap/49.0;client=SfdcInternalAPI/",
            "transactionKey": "000035af-b239-0581-9f14-461e4187de11",
            "commitTimestamp": 1597947935000,
            "recordIds": [
                "0016g00000MLhLeAAL"
            ]
        },
```

```
        "LastModifiedDate": "2020-08-20T18:25:35.000Z",
        "Region__c": "America"
    }
}
```

步驟 1： AppFlow 將 Amazon 設定為作為合Salesforce作夥伴事件來源

要將事件發送到 EventBridge，您首先需要將 Amazon 配置 AppFlow Salesforce為合作夥伴事件來源。

1. 在 Amazon 主 AppFlow控台中，選擇「建立流程」。
2. 在流程詳細資訊區段的流程名稱中，輸入流程的名稱。
3. (選用) 輸入流程的描述，然後選擇下一步。
4. 在來源詳細資訊下，從來源名稱下拉式清單中選擇 Salesforce，然後選擇連接以建立新連線。
5. 在連接至 Salesforce 對話方塊中，選擇 Salesforce 環境的生產或沙盒。
6. 在連線名稱欄位中，輸入連線的唯一名稱，然後選擇繼續。
7. 在 Salesforce 對話方塊中，執行下列動作：
   a. 輸入您的 Salesforce 登入憑證以登錄 Salesforce。
   b. 為 Amazon AppFlow 要處理的資料類型選取Salesforce事件。
8. 在「選擇Salesforce事件」下拉式清單中，選取要傳送至的事件類型 EventBridge。
9. 對於目的地，請選擇 Amazon EventBridge。
10. 選取建立新的合作夥伴事件來源。
11. (選用) 為合作夥伴事件來源指定唯一的後綴。
12. 選擇產生合作夥伴事件來源。
13. 選擇 Amazon S3 儲存貯體來存放大於 256 KB 的事件承載檔案。
14. 在流程觸發器區段中，確保已選取發生事件時執行流程。此設定可確保在發生新的 Salesforce 事件時執行流程。
15. 選擇下一步。
16. 針對欄位對應，請選取直接對應所有欄位。或者，您可以從來源欄位名稱清單中選取感興趣的欄位。

    如需有關欄位映射的詳細資訊，請參閱映射資料欄位。
17. 選擇下一步。

18. (選擇性) 在 Amazon 中設定資料欄位的篩選器 AppFlow。
19. 選擇下一步。
20. 檢閱設定，然後選擇建立流程。

設定流程後，Amazon AppFlow 會建立新的合作夥伴事件來源，然後您必須與帳戶中的合作夥伴事件匯流排建立關聯。

步驟 2：設定 EventBridge 接收Salesforce事件

請確保在遵循本節中的指示之前，先設定從 EventBridge 做為目的地的Salesforce事件觸發的 Amazon AppFlow 流程。

若要設定 EventBridge 接收Salesforce事件

1. 在主控台中開啟 [合作夥伴事件來源] 頁 EventBridge 面。
2. 選取您在步驟 1 中建立的 Salesforce 合作夥伴事件來源。
3. 選擇與事件匯流排建立關聯。
4. 驗證合作夥伴事件匯流排的名稱。
5. 選擇關聯。
6. 在 Amazon 主 AppFlow 控台中，開啟您建立的流程，然後選擇啟用流程。
7. 在主控台中開啟「規則」頁 EventBridge 面。
8. 選擇建立規則。
9. 請輸入規則的唯一名稱。
10. 在定義模式區段中，選擇事件模式。
11. 針對事件比對模式，選取依服務預先定義模式。
12. 針對服務提供者區段，選取所有事件。
13. 針對選取事件匯流排，選擇自訂或合作夥伴事件匯流排。
14. 選取您與 Amazon AppFlow 合作夥伴事件來源相關聯的事件匯流排。
15. 針對 「選取目標」，選擇規則執行時要執行的 AWS 服務。一個規則至多可有 5 個目標。
16. 選擇建立。

目標服務會接收為您的帳戶設定的所有 Salesforce 事件。若要篩選事件或將某些事件傳送至不同的目標，您可以使用事件模式下基於內容的篩選。

> **Note**
>
> 對於大於 256KB 的事件，Amazon AppFlow 不會將完整事件發送到。 EventBridge相反地，Amazon 會將事件 AppFlow 放入您帳戶中的 S3 儲存貯體，然後將事件以指向 EventBridge Amazon S3 儲存貯體的指標傳送至。您可以使用指標從儲存貯體中獲取完整事件。

調試 Amazon EventBridge 事件交付

事件傳遞問題可能很難識別， EventBridge 提供了一些方法來調試和從事件傳遞失敗中恢復。

主題

- 事件重試政策和使用無效字母佇列

事件重試政策和使用無效字母佇列

有時，事件未成功交付至規則中指定的目標。例如，當目標資源無法使用、 EventBridge 缺少目標資源的權限或網路條件時，就可能會發生這種情況。如果事件因為可重新擷取的錯誤而未成功傳遞至目標，請 EventBridge 重試傳送事件。您可以設定嘗試的時間長度，以及目標的重試政策設定中重新嘗試的次數。預設情況下， EventBridge 重試傳送事件 24 小時和最多 185 次，並具有指數退回和抖動，或隨機延遲。如果事件在所有重試嘗試都用盡之後仍未傳遞，則會捨棄該事件，而且 EventBridge不會繼續處理該事件。若要避免在無法交付至目標後遺失事件，您可以設定無效字母佇列 (DLQ)，並將所有失敗事件傳送至事件，以便稍後處理。

EventBridge DLQ 是標準的 Amazon SQS 佇列， EventBridge 用來儲存無法成功傳遞至目標的事件。建立規則並新增目標時，您可以選擇是否使用 DLQ。設定 DLQ 時，您可以保留任何未成功交付的事件。然後，您可以解決導致事件交付失敗的問題，並在稍後處理事件。

事件錯誤以不同的方式進行處理。有些事件會捨棄或傳送至 DLQ，而不會嘗試任何重試。例如，針對由於缺少對目標的許可或目標資源不再存在而導致的錯誤，所有重試嘗試都會失敗，直到採取動作解決基礎問題為止。如果您有的話，請直接 EventBridge 將這些事件傳送至 DLQ，而不是重試。

當事件交付失敗時，會將事件 EventBridge 發佈到 Amazon 指 CloudWatch 標，指出目標`invocation`失敗。如果您使用 DLQ，則會將其他指標傳送至 CloudWatch包括`InvocationsSentToDLQ`和`InvocationsFailedToBeSentToDLQ`。

DLQ 中的每則訊息都會包含下列自訂屬性：

- `RULE_ARN`
- `TARGET_ARN`
- `ERROR_CODE`

  以下是 DLQ 可傳回的錯誤代碼範例：

  - `CONNECTION_FAILURE`
  - `CROSS_ACCOUNT_INGESTION_FAILED`
  - `CROSS_REGION_INGESTION_FAILED`
  - `ERROR_FROM_TARGET`
  - `EVENTS_IN_BATCH_REQUEST_REJECTED`
  - `EVENTS_IN_BATCH_REQUEST_REJECTED`
  - `FAILED_TO_ASSUME_ROLE`
  - `INTERNAL_ERROR`

  - INVALID_JSON
  - INVALID_PARAMETER
  - NO_PERMISSIONS
  - NO_RESOURCE
  - RESOURCE_ALREADY_EXISTS
  - RESOURCE_LIMIT_EXCEEDED
  - RESOURCE_MODIFICATION_COLLISION
  - SDK_CLIENT_ERROR
  - THIRD_ACCOUNT_HOP_DETECTED
  - THIRD_REGION_HOP_DETECTED
  - THROTTLING
  - TIMEOUT
  - TRANSIENT_ASSUME_ROLE
  - UNKNOWN
- ERROR_MESSAGE
- EXHAUSTED_RETRY_CONDITION

  可能回傳以下條件：

  - MaximumRetryAttempts
  - MaximumEventAgeInSeconds
- RETRY_ATTEMPTS

下面的影片會超過設定 DLQ：使用無效字母佇列 (DLQ)

主題

- 使用無效字母佇列的考量
- 將許可授予無效字母佇列
- 如何從無效字母佇列重新傳送事件

使用無效字母佇列的考量

設定 DLQ 時，請考慮下列事項。 EventBridge

- 僅支援標準佇列。您無法將 FIFO 佇列用於中的 DLQ。 EventBridge
- EventBridge 在訊息中包含事件中繼資料和訊息屬性，包括：「錯誤碼」、「錯誤訊息」、「用盡重試條件」、「規則 ARN」、「重試嘗試次數」和「目標 ARN」。您可以使用這些值來識別事件和失敗原因。
- 相同帳戶中 DLQ 的許可：
  - 如果您使用主控台將目標新增至規則，並在同一帳戶中選擇 Amazon SQS 佇列，則授予佇列 EventBridge 存取權的資源型政策會附加至佇列。
  - 如果您使用 EventBridge API 的`PutTargets`操作新增或更新規則的目標，並在同一帳戶中選擇 Amazon SQS 佇列，則必須手動授與所選佇列的許可。如需進一步了解，請參閱將許可授予無效字母佇列。
- 從不同 AWS 帳戶使用 Amazon SQS 佇列的許可。
  - 如果您從主控台建立規則，則不會顯示來自其他帳戶的佇列供您選取。您必須為其他帳戶中的佇列提供 ARN，然後手動附加以資源為基礎的政策，以授予佇列的許可。如需進一步了解，請參閱將許可授予無效字母佇列。
  - 如果您使用 API 建立規則，則必須手動將以資源為基礎的政策附加到另一個帳戶中作為無效字母佇列的 SQS 佇列。如需進一步了解，請參閱將許可授予無效字母佇列。
- 您使用的 Amazon SQS 佇列必須位於建立規則的相同區域。

將許可授予無效字母佇列

當您為規則的目標設定 DLQ 時， EventBridge 會將叫用失敗的事件傳送到選取的 Amazon SQS 佇列。若要成功地將事件傳遞至佇列， EventBridge 必須具有這樣做的權限。當您設定規則的目標並使用 EventBridge 主控台選取 DLQ 時，會自動新增權限。如果您使用 API 建立規則，或使用位於不同 AWS 帳戶的佇列，則必須手動建立以資源為基礎的策略，以授與所需權限，然後將其附加至佇列。

以下以資源為基礎的政策示範如何授與將事件訊息傳送 EventBridge 至 Amazon SQS 佇列的必要許可。此原則範例會授與 EventBridge 服務權限，以便使用`SendMessage`作業將訊息傳送至名為 "MyEventDLQ" 的佇列。佇列必須位於帳戶 123456789012 中的 us-west-2 區域中 AWS 。此`Condition`陳述式只允許來自帳戶 123456789012 中 us-west-2 區域中建立的名稱為 MyTestRule "" 的規則的請求。 AWS

```
{
```

```
  "Sid": "Dead-letter queue permissions",
  "Effect": "Allow",
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "arn:aws:sqs:us-west-2:123456789012:MyEventDLQ",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:events:us-west-2:123456789012:rule/MyTestRule"
    }
  }
}
```

若要將政策附加到佇列，請使用 Amazon SQS 主控台，開啟佇列，然後選擇存取政策並編輯政策。您也可以使用 AWS CLI, 以了解更多信息請參閱Amazon SQS 許可。

如何從無效字母佇列重新傳送事件

您可以以下列兩種方式將訊息從無效字母佇列中移出：

- 避免編寫 Amazon SQS 消費者邏輯 - 將無效字母佇列設為 Lambda 函數的事件來源，以耗盡無效字母佇列。
- 撰寫 Amazon SQS 取用者邏輯 — 使用 Amazon SQS API、 AWS 開發套件或撰寫自訂 AWS CLI 取用者邏輯，以便輪詢、處理和刪除 DLQ 中的訊息。

Amazon EventBridge 事件模式

事件模式擁有與其相符事件相同的結構。規則使用事件模式以選取事件並將事件路由到目標。事件模式與一個事件相符或不相符。

> **Important**
>
> 在中 EventBridge，可以建立可能導致電 higher-than-expected 荷和節流的規則。例如，您可能會不小心建立導致無限迴圈的規則，其中該規則會以遞迴方式觸發而不會結束。假設，您建立的規則可能會偵測到已在 Amazon S3 儲存貯體上變更 ACL，並觸發軟體來將它們變更為所需的狀態。如果未謹慎寫入規則，後續對 ACL 的變更會再次觸發規則，建立無限循環。
> 如需有關撰寫精確規則和事件模式以將此類非預期結果降到最低的指引，請參閱 ??? 和 ???。

以下影片介紹了事件模式的基礎知識：如何篩選事件

主題

- 建立事件模式
- 範例事件和事件模式
- 在 Amazon EventBridge 事件模式中匹配空值和空字符串
- Amazon EventBridge 事件模式中的陣列
- Amazon EventBridge 事件模式中的內容過濾
- 使用 EventBridge 沙箱測試事件模式
- 定義 Amazon EventBridge 事件模式的最佳實務

以下事件顯示了來自 Amazon EC2 的簡單 AWS 事件。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
```

```
  "region": "us-west-1",
  "resources": [
    "arn:aws:ec2:us-west-1:123456789012:instance/i-1234567890abcdef0"
  ],
  "detail": {
    "instance-id": "i-1234567890abcdef0",
    "state": "terminated"
  }
}
```

下列事件模式會處理所有 Amazon EC2 `instance-termination` 事件。

```
{
  "source": ["aws.ec2"],
  "detail-type": ["EC2 Instance State-change Notification"],
  "detail": {
    "state": ["terminated"]
  }
}
```

建立事件模式

若要建立事件模式，您必須指定要進行事件模式比對的事件欄位。僅指定用於比對的欄位。上一個事件模式範例僅提供三個欄位的值：頂層欄位`"source"`和`"detail-type"`，以及`"detail"`物件`"state"`欄位內的欄位。 EventBridge 套用規則時，會忽略事件中的所有其他欄位。

若要事件模式匹配事件，該事件必須包含事件模式中所列的所有欄位名稱。欄位名稱也必須顯示在具有相同巢狀結構的事件中。

當您撰寫事件模式來比對事件時，您可以使用 `TestEventPattern` API 或 `test-event-pattern` CLI 指令，來測試您的模式是否能比對出正確的事件。如需詳細資訊，請參閱TestEventPattern。

相符事件值

在事件模式中，要比對的值位於 JSON 數組中，由方括號 ("["、"]") 包圍，以便您可以提供多個值。例如，若要比對來自 Amazon EC2 的事件 AWS Fargate，或者您可以使用下列模式，該模式與`"source"`欄位值為`"aws.ec2"`或的事件相符`"aws.fargate"`。

```
{
```

```
    "source": ["aws.ec2", "aws.fargate"]
}
```

建立事件模式時的考量

以下是建構事件模式時需要考量的一些注意事項：

- EventBridge 會忽略事件中未包含在事件模式中的欄位。其效果是，對於未出現在事件模式中的字段擁有一個 `"*": "*"` 萬用字元。
- 事件模式比對的值遵循 JSON 規則。您可以加入括在引號 (") 中的字串,、數字和關鍵字 `true`、`false`、`null`。
- 對於字符串， EventBridge 使用精確 character-by-character 匹配而不折疊大小寫或任何其他字符串規範化。
- 對於數字， EventBridge 使用字符串表示。例如，300、300.0 和 3.0e2 不會被視為相等。
- 如果為相同的 JSON 欄位指定了多個模式，則 EventBridge 只會使用最後一個模式。
- 請注意，當 EventBridge 編譯事件模式以供使用時，它會使用 dot (.) 作為連接字元。

  這表示 EventBridge 會將下列事件模式視為相同：

  ```
  ## has no dots in keys
  { "detail" : { "state": { "status": [ "running" ] } } }

  ## has dots in keys
  { "detail" : { "state.status": [ "running" ] } }
  ```

  並且這兩種事件模式都將符合以下兩個事件：

  ```
  ## has no dots in keys
  { "detail" : { "state": { "status": "running" } } }

  ## has dots in keys
  { "detail" : { "state.status": "running"  } }
  ```

  > **Note**
  >
  > 這描述了當前的 EventBridge 行為，不應該依賴於不改變。

- 包含重複欄位的事件模式無效。如果樣式包含重複的欄位，則 EventBridge 僅考慮最終欄位值。

例如，以下事件模式將匹配相同的事件：

```
## has duplicate keys
{
  "source": ["aws.s3"],
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail":  {
      "eventSource": ["s3.amazonaws.com"],
      "eventSource": ["sns.amazonaws.com"]
  }
}

## has unique keys
{
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail": { "eventSource": ["sns.amazonaws.com"] }
}
```

並 EventBridge 將以下兩個事件視為相同的事件：

```
## has duplicate keys
{
  "source": ["aws.s3"],
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail":  [
    {
      "eventSource": ["s3.amazonaws.com"],
      "eventSource": ["sns.amazonaws.com"]
    }
  ]
}

## has unique keys
{
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail": [
    { "eventSource": ["sns.amazonaws.com"] }
  ]
```

```
}
```

> **Note**
>
> 這描述了當前的 EventBridge 行為，不應該依賴於不改變。

用於事件模式的比較操作

以下是中所有可用比較運算子的摘要 EventBridge。

比較運算子僅能在分葉節點上運作，除了 `$or` 和 `anything-but`。

Comparison (比較)	範例	Rule syntax (規則語法)
及	位置為「紐約」，日期是「週一」	`"Location": [ "New York" ], "Day": ["Monday"]`
無論如何，但	State 是除了「初始化」之外的任何值。	`"state": [ { "anything-but": "initializing" } ]`
無論如何，但（開始於）	地區不在美國。	`"Region": [ { "anything-but": {"prefix": "us-" } } ]`
任何事情，但（以結束）	FileName 不會以 .png 副檔名結尾。	`"FileName": [ { "anything-but": { "suffix": ".png" } } ]`
任何事情-但（忽略大小寫）	State 是除「初始化」或任何其他外殼變化之外的任何值，例如「初始化」。	`"state": : [{ "anything-but": { "equals-ignore-case": "initializing" }}]}`
開頭為	地區位於美國。	`"Region": [ {"prefix": "us-" } ]`

Comparison (比較)	範例	Rule syntax (規則語法)
開頭為 (忽略大小寫)	服務名稱以字母「eventb」開頭，無論大小寫如何。	`{"service" : [{ "prefix": { "equals-ignore-case": "eventb" }}]}`
空白	LastName 是空的。	`"LastName": [""]`
等於	名稱為「Alice」	`"Name": [ "Alice" ]`
等於 (忽略大小寫)	名稱為「Alice」	`"Name": [ { "equals-ignore-case": "alice" } ]`
結尾為	FileName 以 .png 副檔名結束	`"FileName": [ { "suffix": ".png" } ]`
以 (忽略大小寫) 結束	服務名稱以字母「tbridge」或任何其他大小寫變化結尾，例如「TBRIDGE」。	`{"service" : [{ "suffix": { "equals-ignore-case": "tBridge" }}]}`
存在	ProductName 存在	`"ProductName": [ { "exists": true } ]`
不存在	ProductName 不存在	`"ProductName": [ { "exists": false } ]`
Not	天氣是「下雨」以外的任何天氣	`"Weather": [ { "anything-but": [ "Raining" ] } ]`
Null	UserID 為 Null 值	`"UserID": [ null ]`
數字 (等於)	價格為 100	`"Price": [ { "numeric": [ "=", 100 ] } ]`
數字 (範圍)	價格大於 10，且小於或等於 20	`"Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ]`
或	PaymentType 是「信用卡」或「借記卡」	`"PaymentType": [ "Credit", "Debit"]`

Comparison (比較)	範例	Rule syntax (規則語法)
或 (多個欄位)	位置為「紐約」，或日期是「週一」。	`"$or": [ { "Location": [ "New York" ] }, { "Day": [ "Monday" ] } ]`
萬用字元	任何副檔名為 .png 的檔案，位於“dir”資料夾內	`"FileName": [ { "wildcard": "dir/*.png" } ]`

範例事件和事件模式

您可以使用所有 JSON 資料類型和值來比對事件。下列範例顯示事件以及符合這些事件的事件模式。

欄位比對

您可以在一個字段的值進行比對。請考量下列 Amazon EC2 Auto Scaling 事件。

```
{
  "version": "0",
  "id": "3e3c153a-8339-4e30-8c35-687ebef853fe",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2015-11-11T21:31:47Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "",
    "responseElements": null
  }
}
```

針對上一個事件，您可以使用 `"responseElements"` 欄位進行比對。

```
{
  "source": ["aws.autoscaling"],
  "detail-type": ["EC2 Instance Launch Successful"],
  "detail": {
   "responseElements": [null]
```

```
  }
}
```

值比對

請考量下面的 Amazon Macie 事件，這是經截斷的事件。

```
{
  "version": "0",
  "id": "0948ba87-d3b8-c6d4-f2da-732a1example",
  "detail-type": "Macie Finding",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2021-04-29T23:12:15Z",
  "region":"us-east-1",
  "resources": [

  ],
  "detail": {
    "schemaVersion": "1.0",
    "id": "64b917aa-3843-014c-91d8-937ffexample",
    "accountId": "123456789012",
    "partition": "aws",
    "region": "us-east-1",
    "type": "Policy:IAMUser/S3BucketEncryptionDisabled",
    "title": "Encryption is disabled for the S3 bucket",
    "description": "Encryption is disabled for the Amazon S3 bucket. The data in the
 bucket isn’t encrypted
        using server-side encryption.",
    "severity": {
        "score": 1,
        "description": "Low"
    },
    "createdAt": "2021-04-29T15:46:02Z",
    "updatedAt": "2021-04-29T23:12:15Z",
    "count": 2,
.
.
.
```

下列事件模式會符合嚴重性分數為 1 且計數為 2 的任何事件。

```
{
```

```
  "source": ["aws.macie"],
  "detail-type": ["Macie Finding"],
  "detail": {
    "severity": {
      "score": [1]
    },
    "count":[2]
  }
}
```

在 Amazon EventBridge 事件模式中匹配空值和空字符串

> **Important**
>
> 在中 EventBridge，可以建立可能導致電 higher-than-expected 荷和節流的規則。例如，您可能會不小心建立導致無限迴圈的規則，其中該規則會以遞迴方式觸發而不會結束。假設，您建立的規則可能會偵測到已在 Amazon S3 儲存貯體上變更 ACL，並觸發軟體來將它們變更為所需的狀態。如果未謹慎寫入規則，後續對 ACL 的變更會再次觸發規則，建立無限循環。
> 如需有關撰寫精確規則和事件模式以將此類非預期結果降到最低的指引，請參閱 ??? 和 ???。

您可以建立符合具有 Null 值或空字串的事件中欄位的事件模式。請考量下列範例事件。

查看最佳做法，以避免高於預期的費用和限流

```
{
  "version": "0",
  "id": "3e3c153a-8339-4e30-8c35-687ebef853fe",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2015-11-11T21:31:47Z",
  "region": "us-east-1",
  "resources": [
   ],
  "detail": {
    "eventVersion": "",
    "responseElements": null
   }
}
```

若要符合 `eventVersion` 的值為空字串的事件，請使用下列符合先前事件的事件模式。

```
{
  "detail": {
     "eventVersion": [""]
  }
}
```

若要符合 `responseElements` 的值為 Null 的事件，請使用下列符合先前事件的事件模式。

```
{
  "detail": {
     "responseElements": [null]
  }
}
```

> **Note**
>
> 在模式比對中，Null 值和空字串不是可交換的。符合空字符串的事件模式與 null 的值不匹配。

Amazon EventBridge 事件模式中的陣列

事件模式中每個字段的值是包含一個或多個值的陣列。如果陣列中的任何值與事件中的值相符，則事件模式會與事件相符。如果事件中的值為陣列，則當事件模式陣列和事件陣列的交集為非空交集時的事件模式相符。

> **Important**
>
> 在中 EventBridge，可以建立可能導致電 higher-than-expected 荷和節流的規則。例如，您可能會不小心建立導致無限迴圈的規則，其中該規則會以遞迴方式觸發而不會結束。假設，您建立的規則可能會偵測到已在 Amazon S3 儲存貯體上變更 ACL，並觸發軟體來將它們變更為所需的狀態。如果未謹慎寫入規則，後續對 ACL 的變更會再次觸發規則，建立無限循環。
> 如需有關撰寫精確規則和事件模式以將此類非預期結果降到最低的指引，請參閱 ??? 和 ???。

例如，請考量包含下列欄目的事件模式。

```
"resources": [
   "arn:aws:ec2:us-east-1:123456789012:instance/i-b188560f",
   "arn:aws:ec2:us-east-1:111122223333:instance/i-b188560f",
   "arn:aws:ec2:us-east-1:444455556666:instance/i-b188560f",
]
```

先前的事件模式符合包含下列欄位的事件，因為模式陣列中的第一個項目符合事件陣列中的第二個項目。

```
"resources": [
   "arn:aws:autoscaling:us-east-1:123456789012:autoScalingGroup:eb56d16b-bbf0-401d-
b893-d5978ed4a025:autoScalingGroupName/ASGTerminate",
   "arn:aws:ec2:us-east-1:123456789012:instance/i-b188560f"
]
```

Amazon EventBridge 事件模式中的內容過濾

Amazon EventBridge 支援使用事件模式進行宣告式內容篩選。透過內容篩選功能，您可以撰寫複雜事件模式，並僅在非常特定的情況下符合事件。例如，您可以在下列情況下建立符合事件的事件模式：

- 事件的欄位位於特定數值範圍內。
- 該事件來自特定的 IP 地址。
- 特定欄位不存在於事件 JSON 中。

> **Important**
>
> 在中 EventBridge，可以建立可能導致電 higher-than-expected 荷和節流的規則。例如，您可能會不小心建立導致無限迴圈的規則，其中該規則會以遞迴方式觸發而不會結束。假設，您建立的規則可能會偵測到已在 Amazon S3 儲存貯體上變更 ACL，並觸發軟體來將它們變更為所需的狀態。如果未謹慎寫入規則，後續對 ACL 的變更會再次觸發規則，建立無限循環。
> 如需有關撰寫精確規則和事件模式以將此類非預期結果降到最低的指引，請參閱 ??? 和 ???。

篩選條件類型

- 前綴相符
- 後綴相符
- 除外相符
- 數值比對
- IP 地址比對
- 存在相符
- 電子quals-ignore-case 匹配
- 使用萬用字元比對
- 具有多個相符項目的複雜範例
- 具有 $or 相符項目的複雜範例

前綴相符

您可以根據事件來源中某個值的前綴來比對事件。您可以使用前綴匹配字符串值。

例如，下面的事件模式將符合該 "time" 欄位以 "2017-10-02" 開始的任何事件，如 "time": "2017-10-02T18:43:48Z"。

```
{
  "time": [ { "prefix": "2017-10-02" } ]
}
```

忽略大小寫的前綴匹配

您還可以匹配前綴值，而不管值開頭的字符的大小寫，結合使用 equals-ignore-case prefix.

例如，下列事件模式會比對service欄位以字元字串開始的任何事件EventB，但也是EVENTBeventb、或這些字元的任何其他大寫。

```
{
  "detail": {"service" : [{ "prefix": { "equals-ignore-case": "EventB" }}]}
}
```

後綴相符

您可以根據事件來源中某個值的後綴來比對事件。您可以使用字串值的後綴比對。

例如，下面的事件模式將符合該 "FileName" 欄位以 .png 檔案副檔名結束的任何事件。

```
{
  "FileName": [ { "suffix": ".png" } ]
}
```

忽略大小寫時的後綴匹配

您還可以匹配後綴值，而不管值結尾的字符的大小寫，結合使用 equals-ignore-case suffix.

例如，下列事件模式會比對FileName欄位以字元字串結束的任何事件.png，.PNG或是這些字元的任何其他大寫。

```
{
  "detail": {"FileName" : [{ "suffix": { "equals-ignore-case": ".png" }}]}
}
```

除外相符

除外相符與規則中提供內容之外的任何項目都相符。

您可以使用與字符串和數字之間的除外相符功能，包括僅包含字串或僅包含數字的清單。

以下事件模式顯示與字符串和數字之間的除外相符。

```
{
  "detail": {
    "state": [ { "anything-but": "initializing" } ]
  }
}

{
  "detail": {
    "x-limit": [ { "anything-but": 123 } ]
  }
}
```

以下事件模式顯示與字符串列表之間的除外相符。

```
{
  "detail": {
    "state": [ { "anything-but": [ "stopped", "overloaded" ] } ]
  }
}
```

以下事件模式顯示與數字列表之間的除外相符。

```
{
  "detail": {
    "x-limit": [ { "anything-but": [ 100, 200, 300 ] } ]
  }
}
```

任何事情-但匹配而忽略大小寫

您也可以結合使用 `equals-ignore-case`與`anything-but`, 以匹配字符串值，而不考慮字符大小寫。

以下事件模式匹配包含`state`字符串「初始化」，但也「初始化」，「初始化」或這些字符的任何其他大寫字符的字段。

```
{
  "detail": {"state" : [{ "anything-but": { "equals-ignore-case": "initializing" }}]}
}
```

您可以結合使用equals-ignore-caseanything-but以匹配值列表，以及：

```
{
  "detail": {"state" : [{ "anything-but": { "equals-ignore-case": ["initializing",
 "stopped"] }}]}
}
```

任何事情-但匹配前綴

以下事件模式顯示了除外相符，其與 "state" 欄位中無前綴 "init" 的事件相符。

> **Note**
>
> 除外相符僅適用於單個前綴，而非列表。

```
{
  "detail": {
    "state": [ { "anything-but": { "prefix": "init" } } ]
  }
}
```

任何事情，但匹配後綴

您可以結合使用suffix以匹配字符串值，而不管字符大小寫如何。anything-but

> **Note**
>
> 任何事情-但匹配只適用於單個後綴，而不是列表。

下列事件模式會比對結尾為之FileName欄位的任何值.txt。

```
{
  "detail": {
    "FileName": [ { "anything-but": { "suffix": ".txt" } } ]
```

```
  }
}
```

數值比對

數值比對適用於 JSON 數字的值。數值比對僅限於 -5.0e9 和 +5.0e9 之間的值，具有 15 位數的精確度，或小數點後六位數。

以下顯示事件模式的數值比對，該模式僅符合所有欄位皆為 True 的事件。

```
{
  "detail": {
    "c-count": [ { "numeric": [ ">", 0, "<=", 5 ] } ],
    "d-count": [ { "numeric": [ "<", 10 ] } ],
    "x-limit": [ { "numeric": [ "=", 3.018e2 ] } ]
  }
}
```

IP 地址比對

您可以針對 IPv4 和 IPv6 地址使用 IP 地址比對。下列事件模式顯示針對 IP 地址的比對，這些 IP 地址以 10.0.0 開始，並以 0 到 255 之間的數字結尾。

```
{
  "detail": {
    "sourceIPAddress": [ { "cidr": "10.0.0.0/24" } ]
  }
}
```

存在相符

存在相符在事件的 JSON 中欄位存在或不存在時發揮作用。

存在相符僅適用於分葉節點。它不適用於中繼節點。

下列事件模式會與任何具有 `detail.state` 欄位的事件相符。

```
{
  "detail": {
    "state": [ { "exists": true  } ]
  }
```

```
}
```

先前的事件模式與下面的事件相符。

```
{
  "version": "0",
  "id": "7bf73129-1428-4cd3-a780-95db273d1602",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "123456789012",
  "time": "2015-11-11T21:29:54Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"],
  "detail": {
    "instance-id": "i-abcd1111",
    "state": "pending"
  }
}
```

先前的事件模式與下面的事件不相符，因為它沒有一個 `detail.state` 欄位。

```
{
  "detail-type": [ "EC2 Instance State-change Notification" ],
  "resources": [ "arn:aws:ec2:us-east-1:123456789012:instance/i-02ebd4584a2ebd341" ],
  "detail": {
    "c-count" : {
       "c1" : 100
    }
  }
}
```

電子quals-ignore-case 匹配

無論大小寫如何，E quals-ignore-case 匹配都適用於字符串值。

下列事件模式符合具有與指定字串相符之 `detail-type` 欄位的任何事件，不論大小寫如何。

```
{
  "detail-type": [ { "equals-ignore-case": "ec2 instance state-change notification" } ]
}
```

先前的事件模式與下面的事件相符。

```
{
  "detail-type": [ "EC2 Instance State-change Notification" ],
  "resources": [ "arn:aws:ec2:us-east-1:123456789012:instance/i-02ebd4584a2ebd341" ],
  "detail": {
    "c-count" : {
       "c1" : 100
    }
  }
}
```

使用萬用字元比對

您可以使用萬用字元 (*) 來比對事件模式中的字串值。

> **Note**
>
> 目前，僅在事件匯流排規則中支援萬用字元。

在事件模式中使用萬用字元時的考量事項：

- 您可以在指定的字串值中指定任意數目的萬用字元，但不支援連續的萬用字元。
- EventBridge 支援使用反斜線字元 (\) 在萬用字元篩選器中指定常值 * 和\ 字元：
  - 該字符串 * 表示常值 * 字元
  - 該字符串 \\ 表示常值 \ 字元

  不支援使用反斜線來逸出其他字元。

萬用字元和事件模式複雜性

使用萬用字元規則的複雜程度有限。如果規則太複雜，則在嘗試建立規則`InvalidEventPatternException`時 EventBridge 傳回。如果您的規則產生此類錯誤，請考慮使用以下指導來降低事件模式的複雜性：

- 減少使用的萬用字元數

  僅有在您真正需要匹配多個可能值的情況下使用萬用字元。例如，請考慮下列事件模式，在此模式下您想要比對相同區域中的事件匯流排：

```
{
"EventBusArn": [ { "wildcard": "*:*:*:*:*:event-bus/*" } ]
}
```

在上述情況下，ARN 的許多部分將直接基於您的事件匯流排所在的區域。因此，如果您使用的是 us-east-1 區域，則仍然與所需值相符的不太複雜的模式可能是以下示例：

```
{
"EventBusArn": [ { "wildcard": "arn:aws:events:us-east-1:*:event-bus/*" } ]
}
```

- 減少萬用字元後出現的重複字元序列

  在使用萬用字元之後多次出現相同的字元序列，會增加處理事件模式的複雜度。重新轉換您的事件模式以最小化重複序列。例如，請考慮下列範例，該範例符合任何使用者的檔案名稱 doc.txt 檔案：

  ```
  {
  "FileName": [ { "wildcard": "/Users/*/dir/dir/dir/dir/dir/doc.txt" } ]
  }
  ```

  如果您知道該 doc.txt 檔案僅會出現在指定的路徑中，則可以透過以下方式減少重複的字元序列：

  ```
  {
  "FileName": [ { "wildcard": "/Users/*/doc.txt" } ]
  }
  ```

具有多個相符項目的複雜範例

您可以將多個相符規則合併為更複雜的事件模式。例如，以下事件模式結合了 anything-but 與 numeric。

```
{
  "time": [ { "prefix": "2017-10-02" } ],
  "detail": {
    "state": [ { "anything-but": "initializing" } ],
    "c-count": [ { "numeric": [ ">", 0, "<=", 5 ] } ],
    "d-count": [ { "numeric": [ "<", 10 ] } ],
    "x-limit": [ { "anything-but": [ 100, 200, 300 ] } ]
  }
```

```
}
```

> **Note**
>
> 在構建事件模式時，如果您多次包含一個鍵，則最後一個參考將是用於評估事件的參考。例如，針對以下模式：
>
> ```
> {
>   "detail": {
>     "location": [ { "prefix": "us-" } ],
>     "location": [ { "anything-but": "us-east" } ]
>   }
> }
> ```
>
> 僅有在評估 `location` 時才會將 `{ "anything-but": "us-east" }` 考慮進去。

具有 $or 相符項目的複雜範例

您還可以建立複雜的事件模式，以檢查是否有跨多個字段的任何字段值相符。如果多個欄位的任何值相符，則使用 `$or` 建立相符的事件模式。

請注意，您可以在您的 `$or` 建構模組中個別欄位的模式比對中包含其他篩選器類型，例如數值比對和陣列。

如果滿足下列任何條件，則下列事件模式會相符：

- `c-count` 欄位大於 0 或小於等於 5。
- `d-count` 欄位小於 10。
- `x-limit` 欄位等於 3.018e2。

```
{
  "detail": {
    "$or": [
      { "c-count": [ { "numeric": [ ">", 0, "<=", 5 ] } ] },
      { "d-count": [ { "numeric": [ "<", 10 ] } ] },
      { "x-limit": [ { "numeric": [ "=", 3.018e2 ] } ] }
    ]
  }
```

```
}
```

> **Note**
>
> 如果使用 $or 導致產生超過 1000 個規則組合，則接受事件模式的 API (例如 PutRule、CreateArchive、UpdateArchive 和 TestEventPattern) 將擲回一個 InvalidEventPatternException。
> 若要判定事件模式中的規則組合數，請乘以來自事件模式中每個 $or 陣列的引數總數。例如，上述模式包含一個帶有三個引數的單個 $or 陣列，因此規則組合的總數也是三個。如果您新增了具有兩個引數的另一個 $or 陣列，則規則組合總計會是六個。

使用 EventBridge 沙箱測試事件模式

規則使用事件模式以選取事件並將事件路由到目標。事件模式擁有與其相符事件相同的結構。事件模式與一個事件相符或不相符。

定義事件模式通常是建立新規則或編輯現有規則的較大程序的一部分。但是 EventBridge，您可以使用中的 Sandbox 快速定義事件模式，並使用範例事件來確認模式符合所需的事件，而無需建立或編輯規則。測試完事件模式後，您可 EventBridge 以選擇直接從沙箱中使用該事件模式建立新規則。

如需有關事件模式的詳細資訊，請參閱 ???。

> **Important**
>
> 在中 EventBridge，可以建立可能導致電 higher-than-expected 荷和節流的規則。例如，您可能會不小心建立導致無限迴圈的規則，其中該規則會以遞迴方式觸發而不會結束。假設，您建立的規則可能會偵測到已在 Amazon S3 儲存貯體上變更 ACL，並觸發軟體來將它們變更為所需的狀態。如果未謹慎寫入規則，後續對 ACL 的變更會再次觸發規則，建立無限循環。
> 如需有關撰寫精確規則和事件模式以將此類非預期結果降到最低的指引，請參閱 ??? 和 ???。

使用 EventBridge 沙箱測試事件模式

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇開發人員資源，然後選取沙盒，然後在沙盒頁面上選擇事件模式標籤。
3. 對於事件來源，請選擇AWS 事件或 EventBridge合作夥伴事件。

4. 在範例事件區段中，選擇您要測試事件模式的範例事件類型。

   可使用以下範例事件類型：

   - AWS 事件 — 從支援 AWS 服務的事件中選取。
   - EventBridge 合作夥伴事件 — 從支援 EventBridge的協力廠商服務 (例如 Salesforce) 發出的事件中選取。
   - 輸入我自己的：以 JSON 文字輸入您自己的事件。

     您也可以使用 AWS 或合作夥伴事件作為建立自訂事件的起點。

     1. 選取AWS 事件或EventBridge 合作夥伴活動。
     2. 使用範例事件下拉式清單，選取要用作自訂事件起點的事件。

        EventBridge 顯示範例事件。
     3. 選取複製。
     4. 針對事件類型選取輸入我自己的。
     5. 刪除 JSON 編輯窗格中的範例事件結構，並將 AWS 或夥伴事件貼到其位置。
     6. 編輯事件 JSON 以建立您自己的範例事件。
5. 選擇建立方法。您可以從 EventBridge 結構描述或範本建立事件模式，也可以建立自訂事件模式。

   Existing schema

   若要使用現有的 EventBridge 結構描述來建立事件模式，請執行下列動作：

   1. 在建立方法區段中，針對方法，選取使用結構描述。
   2. 在事件模式區段中，針對結構描述類型，選取從結構描述登錄檔選取結構描述。
   3. 針對結構描述登錄檔，選擇下拉式方塊，然後輸入結構描述登錄檔的名稱，例如 `aws.events`。您也可以從出現的下拉式清單中選取選項。
   4. 針對結構描述，選擇下拉式方塊，然後輸入要使用的結構描述名稱。例如 `aws.s3@ObjectDeleted`。您也可以從出現的下拉式清單中選取選項。
   5. 在模型區段中，選擇任何屬性旁的編輯按鈕以開啟其屬性內容。視需要設定關係與值欄位，然後選擇設定以儲存屬性。

      > **Note**
      >
      > 如需有關屬性定義的資訊，請選擇屬性名稱旁邊的資訊圖示。如需有關如何在事件中設定屬性內容的參考資料，請開啟屬性內容對話方塊的註記區段。

若要刪除屬性的內容，請選擇該屬性的編輯按鈕，然後選擇清除。

6. 選擇在 JSON 中產生事件模式，以 JSON 文字產生並驗證您的事件模式。
7. 若要根據您的測試模式測試範例事件，請選擇測試模式。

   EventBridge 顯示一個訊息方塊，說明您的範例事件是否符合事件模式。

   您也可以選擇以下其中一個選項：

   - 複製：將事件模式複製到設備的剪貼板。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。

Custom schema

若要撰寫自訂結構描述並將其轉換為事件模式，請執行下列動作：

1. 在建立方法區段中，針對方法，選擇使用結構描述。
2. 在事件模式區段中，針對結構描述類型，選擇輸入結構描述。
3. 將結構描述輸入文字方塊。您必須將結構描述格式化為有效的 JSON 文字。
4. 在模型區段中，選擇任何屬性旁的編輯按鈕以開啟其屬性內容。視需要設定關係與值欄位，然後選擇設定以儲存屬性。

   > **Note**
   >
   > 如需有關屬性定義的資訊，請選擇屬性名稱旁邊的資訊圖示。如需有關如何在事件中設定屬性內容的參考資料，請開啟屬性內容對話方塊的註記區段。
   > 若要刪除屬性的內容，請選擇該屬性的編輯按鈕，然後選擇清除。

5. 選擇在 JSON 中產生事件模式，以 JSON 文字產生並驗證您的事件模式。
6. 若要根據您的測試模式測試範例事件，請選擇測試模式。

   EventBridge 顯示一個訊息方塊，說明您的範例事件是否符合事件模式。

   您也可以選擇以下其中一個選項：

   - 複製：將事件模式複製到設備的剪貼板。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。

Event pattern

若要以 JSON 格式撰寫自訂事件模式，請執行下列動作：

1. 在建立方法區段中，針對方法，選擇自訂模式 (JSON 編輯器)。
2. 針對事件模式，請在 JSON 格式文字中輸入您的自訂事件模式。
3. 若要根據您的測試模式測試範例事件，請選擇測試模式。

   EventBridge 顯示一個訊息方塊，說明您的範例事件是否符合事件模式。

   您也可以選擇以下其中一個選項：

   - 複製：將事件模式複製到設備的剪貼板。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。
   - 事件模式表單：打開模式生成器的事件模式。如果圖樣無法依原樣在「模式產生器」中呈現，則會在開啟「圖樣產生器」之前發 EventBridge 出警告。

6. (選用) 若要使用此事件模式建立規則，並將規則指派給特定事件匯流排，請選擇使用模式建立規則。

   EventBridge 帶您進入建立規則的步驟 1，您可以使用此步驟建立規則，並將其指派給您選擇的事件匯流排。

   請注意，步驟 2 - 構建事件模式包含您已經指定的事件模式信息，以及您可以接受或更新的事件模式信息。

   如需如何建立規則的詳細資訊，請參閱 ???。

定義 Amazon EventBridge 事件模式的最佳實務

以下是在事件匯流排規則中定義事件模式時應考慮的一些最佳實務。

避免編寫無限循環

在中 EventBridge，可以建立導致無限迴圈的規則，其中規則會重複觸發。例如，規則可能會偵測到已在 S3 儲存貯體上變更 ACL，並觸發軟體來將它們變更為所需的狀態。如果未謹慎寫入規則，後續對 ACL 的變更會再次觸發規則，建立無限循環。

為了避免這些問題，請盡可能精確地編寫規則的事件模式，以便它們僅匹配您實際想要傳送至目標的事件。在上述範例中，您會建立事件模式來比對事件，以便觸發的動作不會重新觸發相同的規則。例如，在規則中建立一個事件模式，該模式僅會在 ACL 處於錯誤狀態時才符合事件，而不是在任何變更之後。如需詳細資訊，請參閱 ??? 及 ???。

無限循環可能會快速引發較預期還高的費用。這也可能導致限流和延遲事件交付。您可以監控調用率的上限，以便在數量意外峰值時發出警告。

使用預算編列在費用超過您指定的限制時提醒您。如需詳細資訊，請參閱在符合預算的情形下管理成本。

使事件模式盡可能精確

您的事件模式越精確，它就越有可能僅與您實際想要的事件相符，並且在將新事件新增至事件來源或更新現有事件以包含新屬性時避免意外相符。

事件模式可以包含符合下列項目的篩選條件：

- 有關事件的事件中繼資料，例如 `source`、`detail-type`、`account` 或 `region`。
- 事件資料，這是 `detail` 對象內的字段。
- 事件內容，或者 `detail` 物件內欄位的實際值。

大多數模式都很簡單，例如僅指定 `source` 和 `detail-type` 篩選條件。但是， EventBridge 模式包括靈活地篩選事件的任何鍵或值。此外，您可以套用內容篩選條件，例如 `prefix` 和 `suffix` 篩選條件，以改善模式的精確度。如需詳細資訊，請參閱 ???。

將事件來源和詳細資料類型指定為篩選條件

您可以使用 `source` 和 `detail-type` 中繼資料欄位讓事件模式更精確，以減少產生無限迴圈和比對不需要的事件。

當您需要符合兩個或多個字段中的特定值時，請使用 `$or` 比較運算子，而不是在單個值陣列中列出所有可能的值。

對於透過傳送的事件 AWS CloudTrail，我們建議您使用`eventName`欄位作為篩選器。

下列事件模式範例符合`CreateQueue`或`SetQueueAttributes`來自 Amazon 簡單佇列服務，`CreateKey`或來自該 AWS Key Management Service 服務的`DisableKeyRotation`事件。

```
{
  "detail-type": ["AWS API Call via CloudTrail"],
  "$or": [{
      "source": [
        "aws.sqs"
        ],
      "detail": {
        "eventName": [
          "CreateQueue",
          "SetQueueAttributes"
        ]
      }
    },
    {
      "source": [
        "aws.kms"
        ],
      "detail": {
        "eventName": [
          "CreateKey",
          "DisableKeyRotation"
        ]
      }
    }
  ]
}
```

將帳戶和區域指定為篩選條件

在您的事件模式中包含 `account` 和 `region` 欄位，有助於限制跨帳戶或跨區域事件比對。

指定內容篩選條件

基於內容的篩選可以幫助改善事件模式的精確度，同時仍將事件模式的長度保持在最小。例如，根據數值範圍進行比對可能會很有幫助，而不是列出所有可能的數值。

如需詳細資訊，請參閱 ???。

將事件模式的範圍設定為考量事件來源更新

建立事件模式時，您應該考慮事件結構描述和事件網域可能會隨著時間的推移而發展和擴展。同樣地，讓您的事件模式盡可能精確地協助您在事件來源變更或擴充時限制非預期的相符項目。

例如，假設您要比對來自發佈付款相關事件的新微型服務的事件。起始時，服務會使用網域 `acme.payments`，並發佈單一事件 `Payment accepted`：

```
{
  "detail-type": "Payment accepted",
  "source": "acme.payments",
  "detail": {
    "type": "credit",
    "amount": "100",
    "date": "2023-06-10",
    "currency": "USD"
    }
  }
}
```

此時，您可以建立與付款接受事件相符的簡單事件模式：

```
{ “source” : “acme.payments” }
```

但是，假設服務稍後引入了拒絕付款的新事件：

```
{
  "detail-type": "Payment rejected",
  "source": "acme.payments",
  "detail": {
  }
}
```

在這種情況下，您建立的簡單事件模式現在將 `Payment accepted` 與 `Payment rejected` 事件匹配。 EventBridge 將這兩種類型的事件路由到指定的目標進行處理，這可能會導致處理失敗和額外的處理成本。

若要將事件模式範圍限定為僅 `Payment accepted` 事件，您至少需要指定 `source` 和 `detail-type`：

```
{
  "detail-type": "Payment accepted",
  "source": "acme.payments"
  }
}
```

您也可以在事件模式中指定帳戶與區域，以便在跨帳戶或跨區域事件符合此規則時進一步限制。

```
{
  "account": "012345678910",
  "source": "acme.payments",
  "region": "AWS-Region",
  "detail-type": "Payment accepted"
}
```

驗證事件模式

為了確保規則符合所需的事件，我們強烈建議您驗證您的事件模式。您可以使用 EventBridge 控制台或 API 驗證您的事件模式：

- 在 EventBridge 主控台中，您可以在建立規則的過程中建立和測試事件模式，也可以使用沙箱分別建立和測試事件模式。
- 您可以使用TestEventPattern動作以程式設計方式測試事件模式。

Amazon EventBridge 規則

您可以指定如 EventBridge 何處理傳送至每個事件匯流排的事件。若要這麼做，您可以建立規則。規則會指定要傳送給哪些事件以進行處理的目標。單一規則可以將事件傳送至多個目標，然後再平行執行。

您可以建立兩種類型的規則：

- 符合事件資料的規則

  您可以根據事件資料準則 (稱為事件模式) 建立符合傳入事件的規則。事件模式定義事件結構和規則比對的欄位。如果事件符合事件模式中定義的準則，則 EventBridge 會將其傳送至您指定的目標。

  如需詳細資訊，請參閱 ???。
- 依排程執行的規則

  您也可以建立以指定間隔將事件傳送至指定目標的規則。例如，若要定期執行 Lambda 函數，您可以建立要按排程執行的規則。

> **Note**
>
> EventBridge 提供 Amazon EventBridge Scheduler，這是一種無伺服器排程器，可讓您從單一中央受管服務建立、執行和管理任務。 EventBridge Scheduler 具有高度可自訂性，並提供比 EventBridge排程規則改善的可擴充性，並提供更廣泛的目標 API 作業和 AWS 服務。我們建議您使用 EventBridge 排程器依排程叫用目標。如需詳細資訊，請參閱 ???。

以下影片介紹了規則的基礎知識：什麼是規則

Amazon EventBridge 管理規則

除了您建立的規則外， AWS 服務還可以在您的 AWS 帳戶中建立和管理這些服務中某些功能所需的 EventBridge 規則。這些稱為受管規則。

當服務建立受管規則時，它也可以建立IAM 原則，以授與該服務建立規則的權限。以此方式建立的 IAM 政策會以資源層級許可來限縮範圍，而只允許建立必要的規則。

您可以使用強制刪除選項刪除受管規則，但只有在確定其他服務不再需要該規則時，才應刪除這些規則。否則，刪除受管規則會導致倚賴此規則的功能停止運作。

建立回應事件的 Amazon EventBridge 規則。

若要對 Amazon EventBridge 接收到的事件採取行動，您可以建立規則。當事件與您規則定義的事件模式相符時，EventBridge 將事件發送到指定的目標並觸發規則中定義的動作。

下列影片將探討如何建立不同類型的規則，以及如何測試它們：學習規則

遵循以下處理程序，即可建立適用事件回應的 Amazon EventBridge 規則。

建立適用事件回應的規則

下列步驟將逐步引導您如何建立 EventBridge 用來比對事件傳送至指定事件匯流排的規則。

步驟

- 定義規則
- 建置事件模式
- 選取目標
- 設定標籤和檢閱規則

定義規則

首先，輸入規則的名稱和說明以定義規則。您還必須定義事件匯流排，您的規則會在其中尋找與事件模式相符的事件。

若要定義規則的詳細資訊

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入 Name (名稱)，(可選) 輸入規則描述。

   在同一個 AWS 區域 和同一個事件匯流排上，規則不能與另一個規則同名。
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 AWS default event bus (預設事件匯流排)。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。

7. 選擇 Next (下一步)。

建置事件模式

接下來，建置事件模式 若要這麼做，請指定事件來源、選擇事件模式的基礎，然後定義要比對的屬性和值。您也可以在 JSON 中產生事件模式，並根據範例事件進行測試。

若要建置事件模式

1. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。
2. (可選) 在範例事件區段中，選擇您要測試事件模式的範例事件類型。

   可使用以下範例事件類型：

   - AWS 事件：從支援 AWS 服務 發出的事件中選取。
   - EventBridge 合作夥伴事件：從支援 EventBridge 的第三方服務 (例如 Salesforce) 所發出的事件中選取。
   - 輸入我自己的：以 JSON 文字輸入您自己的事件。

     您也可以使用 AWS 或合作夥伴事件作為建立自訂事件的起點。

     1. 選取 AWS 活動 或 EventBridge 合作夥伴事件。
     2. 使用範例事件下拉式清單，選取要用作自訂事件起點的事件。

        EventBridge 會顯示範例事件。
     3. 選取複製。
     4. 針對事件類型選取輸入我自己的。
     5. 刪除 JSON 編輯窗格中的範例事件結構，並將 AWS 或合作夥伴事件貼到其位置。
     6. 編輯事件 JSON 以建立您自己的範例事件。
3. 選擇建立方法。您可以從 EventBridge 結構描述或範本建立事件模式，也可以建立自訂事件模式。

   Existing schema

   若要使用現有的 EventBridge 結構描述建立事件模式，請執行下列動作：

   1. 在建立方法區段中，對於方法，選取使用結構描述。
   2. 在事件模式區段中，對於結構描述類型，選取從結構描述登錄檔選取結構描述。

3. 對於結構描述登錄檔，選擇下拉式方塊，然後輸入結構描述登錄檔的名稱，例如 `aws.events`。您也可以從出現的下拉式清單中選取選項。
4. 對於結構描述，選擇下拉式方塊，然後輸入要使用的結構描述名稱。例如：`aws.s3@ObjectDeleted`。您也可以從出現的下拉式清單中選取選項。
5. 在模型區段中，選擇任何屬性旁的編輯按鈕以開啟其屬性內容。視需要設定關係與值欄位，然後選擇設定以儲存屬性。

   > **Note**
   >
   > 如需有關屬性定義的資訊，請選擇屬性名稱旁邊的資訊圖示。如需有關如何在事件中設定屬性內容的參考資料，請開啟屬性內容對話方塊的註記區段。
   > 若要刪除屬性的內容，請選擇該屬性的編輯按鈕，然後選擇清除。

6. 選擇在 JSON 中產生事件模式，以 JSON 文字產生並驗證您的事件模式。
7. (可選) 若要根據您的測試模式測試範例事件，請選擇測試模式。

   EventBridge 會顯示一個訊息方塊，說明您的範例事件是否符合事件模式。

   您也可以選擇以下其中一個選項：

   - 複製：將事件模式複製到設備的剪貼板。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。

Custom schema

若要撰寫自訂結構描述並將其轉換為事件模式，請執行下列動作：

1. 在建立方法區段中，對於方法，選擇使用結構描述。
2. 在事件模式區段中，對於結構描述類型，選擇輸入結構描述。
3. 將結構描述輸入文字方塊。您必須將結構描述格式化為有效的 JSON 文字。
4. 在模型區段中，選擇任何屬性旁的編輯按鈕以開啟其屬性內容。視需要設定關係與值欄位，然後選擇設定以儲存屬性。

   > **Note**
   >
   > 如需有關屬性定義的資訊，請選擇屬性名稱旁邊的資訊圖示。如需有關如何在事件中設定屬性內容的參考資料，請開啟屬性內容對話方塊的註記區段。

若要刪除屬性的內容，請選擇該屬性的編輯按鈕，然後選擇清除。

5. 選擇在 JSON 中產生事件模式，以 JSON 文字產生並驗證您的事件模式。
6. (可選) 若要根據您的測試模式測試範例事件，請選擇測試模式。

   EventBridge 會顯示一個訊息方塊，說明您的範例事件是否符合事件模式。

   您也可以選擇以下其中一個選項：

   - 複製：將事件模式複製到設備的剪貼板。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。

Event pattern

若要撰寫 JSON 格式的自訂事件模式，請執行下列動作：

1. 在建立方法區段中，對於方法，選擇自訂模式 (JSON 編輯器)。
2. 對於事件模式，請在 JSON 格式文字中輸入您的自訂事件模式。
3. (可選) 若要根據您的測試模式測試範例事件，請選擇測試模式。

   EventBridge 會顯示一個訊息方塊，說明您的範例事件是否符合事件模式。

   您也可以選擇以下其中一個選項：

   - 複製：將事件模式複製到設備的剪貼板。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。
   - 事件模式表單：打開模式生成器的事件模式。如果模式無法依原樣在「模式產生器」中呈現，EventBridge 會在開啟「模式產生器」之前警告您。

4. 選擇 Next (下一步)。

選取目標

選擇一或多個目標以接收符合指定模式的事件。目標可以包括 EventBridge 事件匯流排、EventBridge API 目的地，包括 SaaS 合作夥伴 (例如 Salesforce) 或其他 AWS 服務。

若要選取目標

1. 對於目標類型，請選擇下列其中一個：

Event bus

若要選取 EventBridge 事件匯流排，請選取事件 EventBridge 事件匯流排，然後執行下列動作：

- 若要在同一個 AWS 區域 下使用與此規則相同的事件匯流排，請執行下列動作：
  1. 選擇相同帳戶和地區中的事件匯流排。
  2. 對於目標的事件匯流排，選擇下拉式方塊並輸入事件匯流排的名稱。您也可以從下拉式清單中選取事件匯流排。

     如需更多詳細資訊，請參閱 ???。
- 若要在不同 AWS 區域 或帳戶中使用事件匯流排，請按照下列規則：
  1. 選取不同帳戶或地區中的事件匯流排。
  2. 對於作為目標的事件匯流排，輸入您要使用的事件匯流排的 ARN。

     如需詳細資訊，請參閱：
     - ???
     - ???

API destination

若要使用 EventBridge API 目的地，請選取 EventBridge API 目的地，然後執行下列其中一個動作：

- 若要使用現有的 API 目的地，請選取使用現有的 API 目的地。然後從下拉式清單中選取 API 目標。
- 若要建立新的 API 目的地，請選取建立新的 API 目的地。接下來，為目的地提供以下詳細資訊：
  - Name (名稱)：輸入目的地名稱。

    名稱在您的 AWS 帳戶 內必須是獨一無二的。名稱長度最長可達 64 個字元。有效字元為 A-Z、a-z、0-9 和 - (連字號)。
  - (選用) 說明：請輸入目的地的說明。

    說明最多可有 512 個字元。
  - API 目標端點：目標的 URL 端點。

端點 URL 必須以 **`https`** 開頭。您可以包含 * 作為路徑參數萬用字元。您可以從目標的 `HttpParameters` 屬性設置路徑參數。

- HTTP 方法：選取調用端點時使用的 HTTP 方法。
- (選用) 每秒調用速率限制：輸入此目的地每秒可接受的調用數目上限。

  該值必須大於零。依預設，此值設為 300。
- 連線：選擇使用新的或現有的連線：
  - 若要使用現有的連線，請選取使用現有連線，然後從下拉式清單中選取連線。
  - 若要為此目的地建立新連線，請選取建立新連線，然後定義連線的名稱、目的地類型和授權類型。您也可以為此連線新增選擇性的描述。

如需更多詳細資訊，請參閱 ???。

AWS 服務

若要使用 AWS 服務，請選取 AWS 服務，然後執行下列動作：

1. 對於選取目標，請選取 AWS 服務 要用作目標的目標。提供您所選服務所要求的資訊。

> **Note**
>
> 顯示的欄位會因選擇的服務而異。如需目標的詳細資訊，請參閱 EventBridge 控制台中可用的目標。

2. 對於許多目標類型而言，EventBridge 需要許可才能將事件傳送到目標。在這些情況下，EventBridge 可建立執行您的規則所需的 IAM 角色。

   對於 Execution role (執行角色)，執行下列任何一項：

   - 若要為此規則建立新的執行角色：
     a. 選取 Create a new role for this specific resource (為此特定資源建立新角色)。
     b. 輸入此執行角色的名稱，或使用 EventBridge 產生的名稱。
   - 若要針對此規則使用現有的執行角色：
     a. 選取使用現有角色。
     b. 從下拉式清單中輸入或選取要使用的執行角色名稱。
3. (選用) 在其他設定中，指定任何可供您目標類型使用的選擇性設定：

Event bus

(選用) 針對 Dead-letter queue (無效字母佇列)，選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。若與此規則匹配的事件未成功傳送到目標，則 EventBridge 會將其傳送至無效字母佇列。執行下列任意一項：

- 選擇 None (無)，即不使用無效字母佇列。
- 選擇 Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue (選擇當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列)，然後從下拉式清單中選擇要使用的佇列。
- 選擇 Select an Amazon SQS queue in an other AWS account as a dead-letter queue (選擇其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列)，然後輸入要使用的佇列的 ARN。您必須將以資源為基礎政策連接到佇列，而且該佇列授與 EventBridge 向其傳送簡訊的許可。

  如需更多詳細資訊，請參閱 將許可授予無效字母佇列。

API destination

1. (選用) 對於設定目標輸入，請選擇您要如何自訂傳送至目標的文字以進行相符事件。選擇下列其中一項：
   - 符合的事件：EventBridge 會將整個原始來源事件傳送至目標。此為預設值。
   - 相符事件的一部分：EventBridge 只會將原始來源事件的指定部分傳送至目標。

     在指定相符事件的部分下，指定定義您希望 EventBridge 傳送至目標之事件部分的 JSON 路徑。
   - 常數 (JSON 文字)：EventBridge 只會將指定的 JSON 文字傳送至目標。不會傳送原始來源事件的任何部分。

     在在 JSON 中指定常數下，指定您希望 EventBridge 傳送到目標而非事件的 JSON 文字。
   - 輸入轉換器：設定輸入轉換器，以自訂您希望 EventBridge 傳送至目標的文字。如需更多詳細資訊，請參閱 ???。

     a. 選取 Configure input transformer (設定輸入轉換器)。

     b. 按照中的步驟配置輸入轉換器 ???。
2. (選用) 在「重試」政策下，指定 EventBridge 在發生錯誤後如何重試將事件傳送至目標。

- 最長事件保留時間：輸入 EventBridge 保留未處理事件的時間上限 (以小時、分鐘和秒為單位)。預設值為 18 小時。
- 重試嘗試：輸入發生錯誤後，EventBridge 應該重試將事件傳送至目標的次數上限。預設值為 185 次。

3. (選用) 針對 Dead-letter queue (無效字母佇列)，選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。若與此規則匹配的事件未成功傳送到目標，則 EventBridge 會將其傳送至無效字母佇列。執行下列任意一項：
   - 選擇 None (無)，即不使用無效字母佇列。
   - 選擇 Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue (選擇當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列)，然後從下拉式清單中選擇要使用的佇列。
   - 選擇 Select an Amazon SQS queue in an other AWS account as a dead-letter queue (選擇其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列)，然後輸入要使用的佇列的 ARN。您必須將以資源為基礎政策連接到佇列，而且該佇列授與 EventBridge 向其傳送簡訊的許可。

     如需更多詳細資訊，請參閱 將許可授予無效字母佇列。

AWS service

請注意，EventBridge 可能不會顯示指定 AWS 服務的下列所有欄位。

1. (選用) 對於設定目標輸入，請選擇您要如何自訂傳送至目標的文字以進行相符事件。選擇下列其中一項：
   - 符合的事件：EventBridge 會將整個原始來源事件傳送至目標。此為預設值。
   - 相符事件的一部分：EventBridge 只會將原始來源事件的指定部分傳送至目標。

     在指定相符事件的部分下，指定定義您希望 EventBridge 傳送至目標之事件部分的 JSON 路徑。
   - 常數 (JSON 文字)：EventBridge 只會將指定的 JSON 文字傳送至目標。不會傳送原始來源事件的任何部分。

     在在 JSON 中指定常數下，指定您希望 EventBridge 傳送到目標而非事件的 JSON 文字。
   - 輸入轉換器：設定輸入轉換器，以自訂您希望 EventBridge 傳送至目標的文字。如需更多詳細資訊，請參閱 ???。

a. 選取 Configure input transformer (設定輸入轉換器)。

b. 按照中的步驟配置輸入轉換器 ???。

2. (選用) 在「重試」政策下，指定 EventBridge 在發生錯誤後如何重試將事件傳送至目標。
   - 最長事件保留時間：輸入 EventBridge 保留未處理事件的時間上限 (以小時、分鐘和秒為單位)。預設值為 18 小時。
   - 重試嘗試：輸入發生錯誤後，EventBridge 應該重試將事件傳送至目標的次數上限。預設值為 185 次。
3. (選用) 針對 Dead-letter queue (無效字母佇列)，選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。若與此規則匹配的事件未成功傳送到目標，則 EventBridge 會將其傳送至無效字母佇列。執行下列任意一項：
   - 選擇 None (無)，即不使用無效字母佇列。
   - 選擇 Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue (選擇當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列)，然後從下拉式清單中選擇要使用的佇列。
   - 選擇 Select an Amazon SQS queue in an other AWS account as a dead-letter queue (選擇其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列)，然後輸入要使用的佇列的 ARN。您必須將以資源為基礎政策連接到佇列，而且該佇列授與 EventBridge 向其傳送簡訊的許可。

   如需更多詳細資訊，請參閱 將許可授予無效字母佇列。
4. (選用) 選擇 Add another target (新增其他目標)，為此規則新增另一個目標。
5. 選擇 Next (下一步)。

請注意，EventBridge 可能不會顯示指定 AWS 服務的下列所有欄位。

設定標籤和檢閱規則

最後，為規則輸入任何想要的標籤，然後檢閱並建立規則。

若要設定標籤，以及檢閱和建立規則

1. (選用) 為規則輸入一或多個標籤。如需更多詳細資訊，請參閱 Amazon EventBridge 標籤。
2. 選擇 Next (下一步)。
3. 檢閱新規則的詳細資料。若要對區段進行變更，請為要編輯的區段選擇編輯按鈕。

   如果您滿意規則詳細資訊，請選擇建立規則。

使用 Amazon EventBridge 排程器搭配 Amazon EventBridge

Amazon EventBridge 排程器是無伺服器排程器，可讓您從單一受管的中央服務建立、執行及管理任務。使用 EventBridge 排程器，您可以使用週期性模式的 Cron 和 Rate 表達式來建立排程，或設定一次性呼叫。您可以設定彈性的交付時段、定義重試次數上限，以及設定失敗的 API 調用的最長保留時間。

EventBridge 排程器具有高度可自訂性，並透過 EventBridge 排程規則改善可擴展性，提供更廣泛的目標 API 操作和 AWS 服務。我們建議您使用 EventBridge 排程器，依照排程調用目標。

主題

設定執行角色

當您建立新排程時，EventBridge 排程器必須具有代表您調用其目標 API 操作的權限。您可以使用執行角色，授與 EventBridge 排程器這些許可。排程執行角色所連接的許可政策會定義哪些是必要許可。許可是否為必要權限，取決於您希望 EventBridge 排程器調用的目標 API。

您在 EventBridge 排程器主控台建立排程時 (如以下程序所述)，EventBridge 排程器會根據您選取的目標自動設定執行角色。如果您要使用 EventBridge 排程器 SDK、AWS CLI 或 AWS CloudFormation 建立排程，您必須具備現有的執行角色，授與 EventBridge 排程器調用目標所需的許可。如需手動設定排程執行角色的詳細資訊，請參閱《EventBridge 排程器使用者指南》中的設定執行角色。

建立排程

使用主控台建立排程

1. 前往 https://console.aws.amazon.com/scheduler/home 開啟 Amazon EventBridge 排程器。
2. 在排程頁面上，選擇建立排程。
3. 在指定排程詳細資訊頁面的排程名稱和描述區段中，執行以下動作：
   a. 在排程名稱中，輸入排程的名稱，例如：**`MyTestSchedule`**。
   b. (選用) 在描述中，輸入對排程的描述，例如：**`My first schedule`**。

c. 針對排程群組，從下拉式清單中選擇排程群組。如果您沒有群組，請選擇預設值。若要建立排程群組，請選擇建立自己的排程。

您可以使用排程群組，為不同群組的排程加上標籤。

4. • 選擇排程選項。

頻率	執行此作業...
一次性排程 一次性排程只會在您指定的日期與時間調用目標一次。	針對日期和時間執行以下動作： • 依 YYYY/MM/DD 格式輸入有效日期。 • 依 hh:mm 格式輸入時間戳記 (24 小時)。 • 針對時區選擇時區。
週期性排程 週期性排程會依您指定的頻率，使用 cron 或 Rate 運算式調用目標。	a. 在排程模式中，執行下列其中一項動作： • 若要使用 Cron 運算式定義排程，請選擇 Cron 排程，然後輸入 Cron 運算式。 • 若要使用 Rate 表達式定義排程，請選擇 Rate 排程，然後輸入 Rate 表達式。 如需 Cron 和 Rate 運算式的詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的 EventBridge 排程器上的排程類型。

頻率	執行此作業...	
	b. 對於彈性時段，選擇關閉可關閉此選項，或者也能選擇其中一個預先定義的時間範圍。例如，如果您選擇 15 分鐘並設定週期性排程，每小時調用目標一次，則排程會在每小時一開始的 15 分鐘內執行。	

5. (選用) 如果您在上一個步驟中選擇週期性排程，請在時間範圍區段執行以下動作：

   a. 針對時區選擇時區。

   b. 對於開始日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。

   c. 對於結束日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。

6. 選擇下一步。

7. 在選取目標頁面上，選擇 EventBridge 排程器調用的 AWS API 操作：

   a. 針對目標 API，選擇範本化目標。

   b. 選擇 Amazon EventBridge PutEvents。

   c. 在PutEvents下，指定下列項目：

      - 對於 EventBridge 事件匯流排，從下拉式選單中選擇事件匯流排。例如：**`default`**。

        您也可以選擇建立新的事件匯流排，在 EventBridge 主控台中建立新的事件匯流排。

      - 如需詳細資訊類型，請輸入您要比對之事件的詳細資訊類型。例如：**`Object Created`**。

      - 在來源中，輸入作為事件來源的服務名稱。

        對於 AWS 服務事件，請將服務前置詞指定為來源。請勿包含 `aws.` 前綴。例如，對於 Amazon S3 事件，請輸入 **`s3`**。

        若要判斷服務的前置詞，請參閱服務授權參考中的條件索引鍵資料表。如需有關來源與詳細資訊類型事件值的詳細資訊，請參閱 ???。

- (選用)：對於詳細資訊，請輸入事件模式，以進一步篩選事件 Bridge 排程器傳送至 EventBridge 的事件。

  如需更多詳細資訊，請參閱 ???。

8. 選擇下一步。
9. 在設定頁面執行以下動作：

   a. 若要開啟排程，請在排程狀態底下切換到啟用排程。

   b. 若要設定排程的重試政策，請在重試政策和無效字母佇列 (DLQ) 底下執行以下動作：

   - 切換到重試。
   - 針對事件的最長存留期，輸入 EventBridge 排程器保留未處理事件的最大時數和分鐘數。
   - 時間最長可設為 24 小時。
   - 針對重試次數上限，輸入目標傳回錯誤時，EventBridge 排程器重新嘗試執行排程的次數上限。

     最大值為重試 185 次。

   設定好重試政策後，如果排程無法調用其目標，EventBridge 排程器會重新執行排程。一旦設定此功能，您就必須設定排程的最長保留時間和重試次數。

   c. 選擇 EventBridge 排程器儲存未交付事件的位置。

無效字母佇列 (DLQ) 選項	執行此作業...
不儲存	選擇無。
將事件儲存在您建立排程所使用的同一個 AWS 帳戶	a. 選擇在目前的 AWS 帳戶選取 Amazon SQS 佇列作為 DLQ。 b. 選擇 Amazon SQS 佇列的 Amazon Resource Name (ARN)。
將事件儲存在建立排程所用帳戶以外的 AWS 帳戶	a. 選擇在其他 AWS 帳戶指定 Amazon SQS 佇列作為 DLQ。

無效字母佇列 (DLQ) 選項	執行此作業...	
	b. 輸入 Amazon SQS 佇列的 Amazon Resource Name (ARN)。	

d. 若要使用由客戶管理的金鑰加密您的目標輸入，請在加密底下選擇自訂加密設定 (進階)。

   如果選擇此選項，請輸入現有的 KMS 金鑰 ARN，或選擇建立 AWS KMS key，以導覽至 AWS KMS 控制台。如需 EventBridge 排程器如何加密靜態資料的詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的靜態加密。

e. 若要讓 EventBridge 排程器為您建立新的執行角色，請選擇為此排程建立新角色。接著輸入角色名稱。如果您選擇此選項，EventBridge 排程器會將範本化目標所需的必要許可與角色連接。

10. 選擇下一步。
11. 在檢閱和建立排程頁面上，檢閱排程的詳細資訊。在每個區段中選擇編輯，即可返回該步驟並編輯其詳細資訊。
12. 選擇建立排程。

    您可以在排程頁面檢視新建立和現有的排程。在狀態欄底下，確認您的新排程狀態為已啟用。

相關資源

如需 EventBridge 排程器的詳細資訊，請參閱下列內容：

- EventBridge 排程器使用者指南
- EventBridge 排程器 API 參考
- EventBridge 排程器定價

建立依排程執行的 Amazon EventBridge 規則。

規則可以在回應事件或特定時間間隔執行。例如，若要定期執行 AWS Lambda 函數，您可建立依照排程執行的規則。

> **Note**
>
> Amazon EventBridge 排程器是無伺服器排程器，可讓您從單一受管的中央服務中建立、執行及管理任務。EventBridge 排程器具有高度可自訂性，並透過 EventBridge 排程規則改善可擴展性，提供更廣泛的目標 API 操作和 AWS 服務。
> 我們建議您使用 EventBridge 排程器，依照排程調用目標。如需更多詳細資訊，請參閱 ???。

在 EventBridge 中，您可以建立兩種類型的排程規則：

- 以一般費率執行的規則

  EventBridge 會定期執行這些規則，例如，每 20 分鐘執行一次。

  若要指定排程規則的比率，請定義Rate 表達式。
- 在特定時間執行的規則

  EventBridge 會在特定的時間和日期執行這些規則，例如：上午 8:00 太平洋標準時間為每個月的第一個星期一。

  若要指定排程規則執行的時間和日期，請定義 Cron 表達式。

Rate 表達式的定義較為簡單，而 Cron 表達式則提供詳細的排程控制。例如，透過 cron 表達式，您可以定義一條規則，在每週或每個月的特定某一天中指定的時間執行。相對的，rate 表達式會以固定的頻率執行規則，例如每個小時一次或是每天一次。

所有排程事件都使用 UTC+0 時區，且排程的最小精度為 1 分鐘。

> **Note**
>
> EventBridge 在排程表達式中不提供第二層級的精確度。使用 cron 表達式的最小解析是一分鐘。由於 EventBridge 和目標服務的分散式特性，觸發排程規則與目標服務執行目標資源之間可能會有幾秒鐘的延遲。

下列影片提供排程工作的概觀：使用 EventBridge 建立排程工作

主題

- 建立依排程執行的規則
- Cron 表達式範例
- Rate 表達式參照

建立依排程執行的規則

下列步驟將逐步引導您如何建立依照定期排程執行的 EventBridge 規則。

> **Note**
>
> 您只能使用預設事件匯流排建立排程規則。

步驟

- 定義規則
- 定義排程
- 選取目標
- 設定標籤和檢閱規則

定義規則

首先，輸入規則的名稱和說明以定義規則。

若要定義規則詳細資訊

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules(規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入 Name (名稱)，(可選) 輸入規則描述。

   在同一個 AWS 區域 和同一個事件匯流排上，規則不能與另一個規則同名。
5. 對於Event bus (選取事件匯流排)，選擇 default event bus (預設事件匯流排)。您只能使用預設事件匯流排建立排程規則。
6. 若要在建立規則後立即生效，請確定已啟用在選取的事件匯流排上啟用規則選項。
7. 針對 Rule type(規則類型)，選擇 Schedule(排程)。

此時，您可以選擇繼續建立依排程執行的規則，或使用 Amazon EventBridge 排程器。

8. 選擇您要繼續的方式：

   - 使用 EventBridge 排程器建立排程

     > **Note**
     >
     > EventBridge 排程器是無伺服器排程器，可讓您從單一受管的中央服務建立、執行及管理任務。其提供與事件匯流排和規則無關的一次性和週期性排程功能。EventBridge 排程器具有高度可自訂性，並透過 EventBridge 排程規則改善可擴展性，提供更廣泛的目標 API 操作和 AWS 服務。
     > 我們建議您使用 EventBridge 排程器，依照排程調用目標。如需詳細資訊，請參閱《Amazon EventBridge 排程器 使用者指南》中的什麼是 Amazon EventBridge 排程器？。

     1. 選取在 EventBridge 排程器中繼續

        EventBridge 會開啟「EventBridge 排程器」主控台至建立排程頁面。

     2. 在 EventBridge 排程器主控台中建立排程。

   - 繼續使用 EventBridge 為預設事件匯流排建立排程規則

     1. 選取「繼續」以建立規則。

定義排程

下一步，定義排程模式。

若要定義排程模式

1. 對於「排程」模式，請選擇要在特定時間執行排程，還是以一般費率執行：

   Specific time

   1. 選擇在特定時間執行的精細排程，例如上午 8:00 太平洋標準時間為每個月的第一個星期一。
   2. 對於 Cron 表達式，請指定欄位以定義 EventBridge 應該用來決定何時執行此排程規則的排程運算式。

當您指定完所有欄位之後，EventBridge 會顯示接下來的十個日期，讓 EventBridge 執行此排程規則。您可以選擇以 UTC 或當地時區顯示這些日期。

如需建構 Cron 表達式的詳細資訊，請參閱 ???。

Regular rate

1. 選擇以一般費率執行的排程，例如每 10 分鐘執行一次。
2. 對於Rate 表達式，請指定值和單位欄位，以定義 EventBridge 應執行此排定規則的速率。

   如需建構 Rate 表達式的詳細資訊，請參閱 ???。

2. 選擇 Next (下一步)。

選取目標

選擇一或多個目標以接收符合指定模式的事件。目標可以包括 EventBridge 事件匯流排、EventBridge 接 API 目的地，包括 SaaS 合作夥伴 (例如 Salesforce) 或其他 AWS 服務。

若要選取目標

1. 對於目標類型，請選擇下列其中一個：

   Event bus

   若要選取 EventBridge 事件匯流排，請選取事件 EventBridge 事件匯流排，然後執行下列動作：

   - 若要在同一個 AWS 區域 下使用與此規則相同的事件匯流排，請執行下列動作：
     1. 選擇相同帳戶和地區中的事件匯流排。
     2. 對於目標的事件匯流排，選擇下拉式方塊並輸入事件匯流排的名稱。您也可以從下拉式清單中選取事件匯流排。

        如需更多詳細資訊，請參閱 ???。
   - 若要在不同 AWS 區域 或帳戶中使用事件匯流排，請按照下列規則：
     1. 選擇不同帳戶或地區中的事件匯流排。
     2. 對於作為目標的事件匯流排，輸入您要使用的事件匯流排的 ARN。

如需詳細資訊，請參閱：

- ???
- ???

API destination

若要使用 EventBridge API 目的地，請選取 EventBridge API 目的地，然後執行下列其中一個動作：

- 若要使用現有的 API 目的地，請選取 [使用現有的 API 目的地]。然後從下拉式清單中選取 API 目標。
- 若要建立新的 API 目的地，請選取建立新的 API 目的地。接下來，為目的地提供以下詳細資訊：
  - Name (名稱)：輸入目的地名稱。

    名稱在您的 AWS 帳戶 內必須是獨一無二的。名稱長度最長可達 64 個字元。有效字元為 A-Z、a-z、0-9 和 _ - (連字號)。
  - (選用) 說明：請輸入目的地的說明。

    說明最多可有 512 個字元。
  - API 目標端點：目標的 URL 端點。

    端點 URL 必須以 **`https`** 開頭。您可以包含 **`*`** 作為路徑參數萬用字元。您可以從目標的 `HttpParameters` 屬性設置路徑參數。
  - HTTP 方法：選取調用端點時使用的 HTTP 方法。
  - (選用) 每秒調用速率限制：輸入此目的地每秒可接受的調用數目上限。

    該值必須大於零。依預設，此值設為 300。
  - 連線：選擇使用新的或現有的連線：
    - 若要使用現有的連線，請選取使用現有連線，然後從下拉式清單中選取連線。
    - 若要為此目的地建立新連線，請選取建立新連線，然後定義連線的名稱、目的地類型和授權類型。您也可以為此連線新增選擇性的描述。

如需更多詳細資訊，請參閱 ???。

AWS 服務

若要使用 AWS 服務，請選取 AWS 服務，然後執行下列動作：

1. 對於選取目標，請選取 AWS 服務 要用作目標的目標。提供您所選服務所要求的資訊。

> **Note**
>
> 顯示的欄位會因選擇的服務而異。如需目標的詳細資訊，請參閱 EventBridge 控制台中可用的目標。

2. 對於許多目標類型而言，EventBridge 需要許可才能將事件傳送到目標。在這些情況下，EventBridge 可建立執行您的規則所需的 IAM 角色。

對於 Execution role(執行角色)，執行下列任何一項：

- 若要為此規則建立新的執行角色：
  a. 選擇 Create a new role for this specific resource (為此特定資源建立新角色)。
  b. 輸入此執行角色的名稱，或使用 EventBridge 產生的名稱。
- 若要針對此規則使用現有的執行角色：
  a. 選取 [使用現有角色]。
  b. 從下拉式清單中輸入或選取要使用的執行角色名稱。

3. (選擇性) 在其他設定中，指定任何可供您目標類型使用的選擇性設定：

Event bus

針對 Dead-letter queue (無效字母佇列)，選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。若與此規則匹配的事件未成功傳送到目標，則 EventBridge 會將其傳送至無效字母佇列。執行下列任意一項：

- 選擇 None (無)，即不使用無效字母佇列。
- 選擇 Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue (選擇當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列)，然後從下拉式清單中選擇要使用的佇列。
- 選擇 Select an Amazon SQS queue in an other AWS account as a dead-letter queue (選擇其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列)，然後輸入要使用的佇列的

ARN。您必須將以資源為基礎政策連接到佇列，而且該佇列授與 EventBridge 向其傳送簡訊的許可。

如需更多詳細資訊，請參閱 將許可授予無效字母佇列。

API destination

1. (選用) 對於設定目標輸入，請選擇您要如何自訂傳送至目標的文字以進行相符事件。選擇下列其中一項：
   - 符合的事件：EventBridge 會將整個原始來源事件傳送至目標。此為預設值。
   - 相符事件的一部分：EventBridge 只會將原始來源事件的指定部分傳送至目標。

     在指定相符事件的部分下，指定定義您希望 EventBridge 傳送至目標之事件部分的 JSON 路徑。
   - 常數 (JSON 文字)：EventBridge 只會將指定的 JSON 文字傳送至目標。不會傳送原始來源事件的任何部分。

     在在 JSON 中指定常數下，指定您希望 EventBridge 傳送到目標而非事件的 JSON 文字。
   - 輸入轉換器：設定輸入轉換器，以自訂您希望 EventBridge 傳送至目標的文字。如需更多詳細資訊，請參閱 ???。

     a. 選擇 Configure input transformer (設定輸入轉換器)。

     b. 按照 ??? 中的步驟配置輸入轉換器。
2. (選用) 在「重試」政策下，指定 EventBridge 在發生錯誤後如何重試將事件傳送至目標。
   - 最長事件保留時間：輸入 EventBridge 保留未處理事件的時間上限 (以小時、分鐘和秒為單位)。預設值為 24 小時。
   - 重試嘗試：輸入發生錯誤後，EventBridge 應該重試將事件傳送至目標的次數上限。預設值為 185 次。
3. (選用) 針對 Dead-letter queue (無效字母佇列)，選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。若與此規則匹配的事件未成功傳送到目標，則 EventBridge 會將其傳送至無效字母佇列。執行下列任意一項：
   - 選擇 None (無)，即不使用無效字母佇列。
   - 選擇 Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue (選擇當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列)，然後從下拉式清單中選擇要使用的佇列。

- 選擇 Select an Amazon SQS queue in an other AWS account as a dead-letter queue (選擇其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列)，然後輸入要使用的佇列的 ARN。您必須將以資源為基礎政策連接到佇列，而且該佇列授與 EventBridge 向其傳送簡訊的許可。

  如需更多詳細資訊，請參閱 將許可授予無效字母佇列。

AWS service

請注意，EventBridge 可能不會顯示指定 AWS 服務的下列所有欄位。

1. (選用) 對於設定目標輸入，請選擇您要如何自訂傳送至目標的文字以進行相符事件。選擇下列其中一項：
   - 符合的事件：EventBridge 會將整個原始來源事件傳送至目標。此為預設值。
   - 相符事件的一部分：EventBridge 只會將原始來源事件的指定部分傳送至目標。

     在指定相符事件的部分下，指定定義您希望 EventBridge 傳送至目標之事件部分的 JSON 路徑。
   - 常數 (JSON 文字)：EventBridge 只會將指定的 JSON 文字傳送至目標。不會傳送原始來源事件的任何部分。

     在在 JSON 中指定常數下，指定您希望 EventBridge 傳送到目標而非事件的 JSON 文字。
   - 輸入轉換器：設定輸入轉換器，以自訂您希望 EventBridge 傳送至目標的文字。如需更多詳細資訊，請參閱 ???。

     a. 選擇 Configure input transformer (設定輸入轉換器)。

     b. 按照 ??? 中的步驟配置輸入轉換器。
2. (選用) 在「重試」政策下，指定 EventBridge 在發生錯誤後如何重試將事件傳送至目標。
   - 最長事件保留時間：輸入 EventBridge 保留未處理事件的時間上限 (以小時、分鐘和秒為單位)。預設值為 24 小時。
   - 重試嘗試：輸入發生錯誤後，EventBridge 應該重試將事件傳送至目標的次數上限。預設值為 185 次。
3. (選用) 針對 Dead-letter queue (無效字母佇列)，選擇是否使用標準 Amazon SQS 佇列做為無效字母佇列。若與此規則匹配的事件未成功傳送到目標，則 EventBridge 會將其傳送至無效字母佇列。執行下列任意一項：

- 選擇 None (無)，即不使用無效字母佇列。
- 選擇 Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue (選擇當前 AWS 帳戶中的 Amazon SQS 佇列以用作無效字母佇列)，然後從下拉式清單中選擇要使用的佇列。
- 選擇 Select an Amazon SQS queue in an other AWS account as a dead-letter queue (選擇其他 AWS 帳戶中的 Amazon SQS 佇列做為無效字母佇列)，然後輸入要使用的佇列的 ARN。您必須將以資源為基礎政策連接到佇列，而且該佇列授與 EventBridge 向其傳送簡訊的許可。

  如需更多詳細資訊，請參閱 將許可授予無效字母佇列。

4. (選用) 選擇 Add another target (新增其他目標)，為此規則新增另一個目標。
5. 選擇 Next (下一步)。

設定標籤和檢閱規則

最後，為規則輸入任何想要的標籤，然後檢閱並建立規則。

若要設定標籤，以及檢閱和建立規則

1. (選用) 為規則輸入一或多個標籤。如需更多詳細資訊，請參閱 Amazon EventBridge 標籤。
2. 選擇 Next (下一步)。
3. 檢閱新規則的詳細資料。若要對區段進行變更，請為要編輯的區段選擇編輯按鈕。

   如果您滿意規則詳細資訊，請選擇建立規則。

Cron 表達式範例

Cron 表達式有六個必要欄位，以空格隔開。

語法

```
cron(fields)
```

欄位	Values (數值)	Wildcards (萬用字元)
分鐘	0-59	, - * /

欄位	Values (數值)	Wildcards (萬用字元)
小時	0-23	, - * /
月中的日	1-31	, - * ? / L W
月	1-12 或 JAN-DEC	, - * /
週中的日	1-7 或 SUN-SAT	, - * ? L #
年	1970-2199	, - * /

萬用字元

- , (逗號) 萬用字元包含額外的值。在 Month (月) 欄位，JAN、FEB、MAR 包括 January (一月)、February (二月) 與 March (三月)。
- - (破折號) 萬用字元用於指定範圍。在 Day (日) 欄位，1-15 包含指定月份的 1 至 15 號。
- * (星號) 包含欄位中所有的值。在 Hours (小時) 欄位，* 包含每個小時。您無法在月中的特定一天和週中的特定一天兩個欄位同時使用 *。若您在其中一個欄位使用它，您必須在另一個欄位使用 ?。
- / (斜線) 萬用字元用於指定增量。在 Minutes (分鐘) 欄位，您可以輸入 1/10 指定每十分鐘的間隔，從小時的第一分鐘開始 (例如第 11、第 21、第 31 分鐘等)。
- ? (問號) 萬用字元用於表示不限定任何一個。在 Day-of-month (月中的日) 欄位，您可以輸入 7，如果您不在意這個月的 7 號是星期幾，就可以在 Day-of-month (月中的日) 欄位中輸入 ?。
- L 萬用字元在 Day-of-month (月中的日) 或 Day-of-week (週中的日) 欄位可指定月份或週的最後一天。
- **W** 萬用字元在 Day-of-month (月中的日) 欄位可指定工作日。在 Day-of-month (月中的日) 欄位，**3W** 指定的是月份中最接近第三個工作日的日子。
- # 萬用字元在 Day-of-week (週中的日) 欄位可指定某個月中某週特定日子的特定執行個體。例如，3#2 代表則該月的第二個星期二：3 是指星期二，因為它是每週的第三天，2 指的是一個月內該類型的第二天。

> **Note**
>
> 如果您使用 '#' 字元，則只能在星期幾欄位中定義一個表達式。例如：`"3#1,6#3"` 是無效的，因為它被轉譯為兩個表達式。

限制

- 您無法在同一個 cron 表達式中指定 Day-of-month (月中的日) 和 Day-of-week (週中的日) 欄位。如果您在其中一個欄位指定了數值或 * (星號)，就必須在另一個欄位中使用 ? (問號)。
- 不支援頻率多於 1 分鐘的 Cron 表達式。

範例

使用排程建立規則時，您可以使用下列 cron 字串範例。

分鐘	小時	月中的日	月	週中的日	年	意義
0	10	*	*	?	*	在每天上午 10:00 (UTC+0) 執行
15	12	*	*	?	*	在每天下午 12:15 (UTC+0) 執行
0	18	?	*	MON-FRI	*	在每週一至週五下午 6:00 (UTC+0) 執行
0	8	1	*	?	*	在每個月第 1 天上午 8:00 (UTC+0) 執行
0/15	*	*	*	?	*	每 15 分鐘執行
0/10	*	?	*	MON-FRI	*	在週一至週五每 10 分鐘執行

分鐘	小時	月中的日	月	週中的日	年	意義
0/5	8-17	?	*	MON-FRI	*	在週一至週五上午 8:00 至下午 5:55 (UTC+0) 之間每 5 分鐘執行
0/30	20-2	?	*	MON-FRI	*	週一至週五每 30 分鐘執行一次，從開始日的晚上 10:00 至次日凌晨 2:00 (UTC 世界標準時間) 在星期一早上 (UTC 世界標準時間) 上午 12 點至凌晨 2 點執行。

以下範例會建立在每天下午 12:00 (UTC+0) 執行之規則。

```
aws events put-rule --schedule-expression "cron(0 12 * * ? *)" --name MyRule1
```

以下範例會建立在每天下午 2:05 至 2:35 (UTC+0) 之間執行之規則。

```
aws events put-rule --schedule-expression "cron(5,35 14 * * ? *)" --name MyRule2
```

以下範例會建立規則，並在 2019 年至 2022 年每個月的最後一個週五 UTC+0 時間上午 10:15 執行。

```
aws events put-rule --schedule-expression "cron(15 10 ? * 6L 2019-2022)" --name MyRule3
```

Rate 表達式參照

Rate 表達式在您建立排程事件規則時開始，然後在定義的排程上執行。

Rate 表達式有六個必要欄位，以空格隔開。

語法

```
rate(value unit)
```

value

正數。

單位

時間的單位。所需單位可能不同，若值為 1，則需要 `minute`；若值超過 1，則需要 `minutes`。

有效值：minute | minutes | hour | hours | day | days (分鐘、數分鐘、小時、數小時、天、數天)

限制

如果值等於 1，則單位必須為單數。如果值大於 1，則單位必須為複數。例如，rate (1 hours) 和 rate (5 hour) 為無效，但 rate (1 hour) 和 rate (5 hours) 為有效。

範例

以下範例說明如何透過 AWS CLI `put-rule` 命令使用 rate 表達式。第一個範例會每分鐘觸發規則，下一個範例則會每 5 分鐘觸發一次，第三個範例會每小時觸發一次，最後一個範例則會每天觸發一次。

```
aws events put-rule --schedule-expression "rate(1 minute)" --name MyRule2
```

```
aws events put-rule --schedule-expression "rate(5 minutes)" --name MyRule3
```

```
aws events put-rule --schedule-expression "rate(1 hour)" --name MyRule4
```

```
aws events put-rule --schedule-expression "rate(1 day)" --name MyRule5
```

停用或刪除 Amazon EventBridge 規則

若要停止規則處理事件或按排程執行，您可以刪除或停用規則。下列步驟將逐步引導您如何刪除或停用 EventBridge 規則。

刪除或停用規則

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。

   在 Event bus (事件匯流排) 下，選取與規則相關聯的事件匯流排。
3. 執行下列任意一項：

   a. 若要刪除規則，請選取規則旁的按鈕，然後選擇 Actions (動作)、Delete (刪除)、Delete (刪除)。

      如果規則是受管規則，輸入規則的名稱以表示它是受管規則，而且刪除它可能會使建立該規則的服務停止運作。若要繼續，請輸入規則名稱並選擇 Force delete (強制刪除)。

   b. 若要暫時停用規則，請選取規則旁的按鈕，然後選擇 Disable (停用)、Disable (停用)。

      您無法停用受管規則。

定義 Amazon EventBridge 規則的最佳實務

以下是為事件匯流排建立規則時應考量的一些最佳作法。

為每個規則設定單一目標

雖然您可以為指定規則最多指定五個目標，但是當您為每個規則指定單一目標時，管理規則會比較容易。如果有多個目標需要接收相同的事件集，建議您複製規則以將相同的事件傳遞至不同的目標。這種封裝簡化了規則的維護：如果事件目標的需求隨著時間的推移而發生差異，您可以獨立於其他規則更新每個規則及其事件模式。

設定規則許可

您可以讓取用事件的應用程式元件或服務控制管理自己的規則。客戶採用的一種常見架構方法是使用不同的 AWS 帳戶來隔離這些應用程式元件或服務。若要啟用事件從一個帳戶到另一個帳戶的流程，您必須在一個事件匯流排上建立規則，將事件路由至另一個帳戶中的事件匯流排。您可以讓取用事件的團隊

或服務控制管理自己的規則。您可以透過資源策略為其帳號指定適當的權限來執行此操作。這適用於帳戶和區域。

如需更多詳細資訊，請參閱 ???。

如需資源政策的範例，請參閱 GitHub 上的 Amazon EventBridge 多帳戶設計模式。

監控規則效能

監控您的規則，以確保它們的執行如您所期望：

- 監視遺失資料點或異常的 `TriggeredRules` 指標，可協助您偵測發布者發生重大變更的差異。如需更多詳細資訊，請參閱 ???。
- 針對異常或預期計數上限發出警示，也可協助偵測規則何時符合新事件。當事件發布者（包括 AWS 服務和 SaaS 合作夥伴）在啟用新的使用案例和功能時引入新事件時，可能會發生這種情況。當這些新事件非預期且導致數量高於下游目標的處理速率時，可能會導致事件積壓。

  這種意外事件的處理也可能導致不必要的帳單費用。

  當帳戶超過其每秒服務配額的彙總目標調用時，也可以觸發規則限流。EventBridge 仍會嘗試傳遞受限流規則相符的事件，並重試最多 24 小時，或如目標自訂重試原則中所述。您可以使用 `ThrottledRules` 量度偵測並警示限流規則
- 對於低延遲的使用案例，您也可以使用 `IngestionToInvocationStartLatency` 來監控延遲，這會提供事件匯流排健康狀態的指示。任何長時間超過 30 秒的高延遲時間可能表示服務中斷或規則限流。

使用 Amazon EventBridge 和 AWS Serverless Application Model 範本

您可以在 EventBridge 主控台中手動建置和測試規則，這可在您精簡事件模式時協助開發程序。不過，一旦您準備好部署應用程式，就可以更輕鬆地使用架構，例如 AWS SAM 一致地啟動所有無伺服器資源。

我們將使用此示例應用程序來研究您可以使用 AWS SAM 模板來構建 EventBridge 資源的方式。此範例中的範本 .yaml 檔案是一個 AWS SAM 範本，它會定義四個 AWS Lambda 函數，並顯示兩種不同的方式來整合 Lambda 函數與 EventBridge。

如需此範例應用程式的逐步解說，請參閱 ???。

有兩種方式使用 EventBridge AWS SAM 範本。對於由一個規則叫用一個 Lambda 函數的簡單整合，建議使用組合範本方法。如果您有複雜的路由邏輯，或者連線到 AWS SAM 範本之外的資源，則分離範本方法是更好的選擇。

方法

- 合併範本
- 分離的模板

合併範本

第一種方法會使用 `Events` 屬性來設定 EventBridge 規則。下列範例程式碼會定義調用 Lambda 函數的事件。

> **Note**
>
> 此範例會自動在每個 AWS 帳戶中存在的預設事件匯流排上建立規則。若要將規則與自訂事件匯流排相關聯，您可以將該規則加入至 `EventBusName` 範本。

```
atmConsumerCase3Fn:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: atmConsumer/
    Handler: handler.case3Handler
```

```
    Runtime: nodejs12.x
    Events:
      Trigger:
        Type: CloudWatchEvent
        Properties:
          Pattern:
            source:
              - custom.myATMapp
            detail-type:
              - transaction
            detail:
              result:
                - "anything-but": "approved"
```

這個 YAML 程式碼等同於事件 EventBridge 主控台中的事件模式。在 YAML 中，您只需要定義事件模式，並 AWS SAM 自動建立具有所需許可的 IAM 角色。

分離的模板

在 AWS SAM 中定義 EventBridge 組態的第二種方法中，資源在範本中會更清楚地分開。

1. 首先，您可以定義 Lambda 函數：

   ```
   atmConsumerCase1Fn:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: atmConsumer/
       Handler: handler.case1Handler
       Runtime: nodejs12.x
   ```

2. 接下來，使用 `AWS::Events::Rule` 資源定義規則。屬性定義事件模式，也可以指定目標。您可以明確定義多個目標。

   ```
   EventRuleCase1:
     Type: AWS::Events::Rule
     Properties:
       Description: "Approved transactions"
       EventPattern:
         source:
           - "custom.myATMapp"
         detail-type:
           - transaction
         detail:
   ```

```
        result:
          - "approved"
    State: "ENABLED"
    Targets:
      -
        Arn:
          Fn::GetAtt:
            - "atmConsumerCase1Fn"
            - "Arn"
        Id: "atmConsumerTarget1"
```

3. 最後，定義一個 AWS::Lambda::Permission 資源，該資源授予 EventBridge 調用目標的權限。

```
PermissionForEventsToInvokeLambda:
  Type: AWS::Lambda::Permission
  Properties:
    FunctionName:
      Ref: "atmConsumerCase1Fn"
    Action: "lambda:InvokeFunction"
    Principal: "events.amazonaws.com"
    SourceArn:
      Fn::GetAtt:
        - "EventRuleCase1"
        - "Arn"
```

從 Amazon EventBridge 規則產生 AWS CloudFormation 範本

AWS CloudFormation 透過將基礎架構視為程式碼，讓您以集中且可重複的方式，跨帳戶和區域設定和管理 AWS 資源。CloudFormation 透過讓您建立範本 (定義您要佈建和管理的資源) 來執行此作業。

EventBridge 可讓您從帳戶中現有的規則產生範本，以協助您快速開始開發 CloudFormation 範本。您可以選取要包含在範本中的單一規則或多個規則。然後，您可以使用這些範本作為基礎，以便建立受 CloudFormation 管理的資源堆疊。

如需 CloudFormation 的詳細資訊，請參閱《AWS CloudFormation 使用者指南》。

> **Note**
>
> EventBridge 不會在產生的範本中包含受管理的規則。

您也可以從現有的事件匯流排 (包括事件匯流排所包含的規則) 產生範本。

若要從一或多個規則產生 AWS CloudFormation 範本

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 在選取事件匯流排下，選擇包含您要包含在範本中之規則的事件匯流排。
4. 在規則下，選擇您要包含在產生 AWS CloudFormation 範本中的規則。

   對於單一管道，您也可以選擇規則名稱以顯示該規則的詳細資訊頁面。
5. 選擇CloudFormation 範本，然後選擇您希望 EventBridge 在其中產生範本的格式：JSON 或 YAML。

   EventBridge 會顯示以所選格式產生的範本。
6. EventBridge 可讓您選擇下載範本檔案，或將範本複製到剪貼簿。

   - 選擇立即下載以下載範本檔案。
   - 若要將範本複製剪貼簿，請選擇複製。
7. 若要結束範本，請選擇取消。

根據使用案例的需要自訂 AWS CloudFormation 範本後，您可以使用它在 AWS CloudFormation 中建立堆疊。

使用從 Amazon EventBridge 產生的 CloudFormation 範本時的注意事項

使用從 EventBridge 產生的 CloudFormation 範本時，請考量下列因素：

- EventBridge 不會在產生的範本中包含任何密碼。

  您可以編輯範本以包含範本參數，讓使用者在使用範本建立或更新 CloudFormation 堆疊時，能夠指定密碼或其他敏感資訊。

  此外，使用者可以使用 Secrets Manager 在所需區域中建立密碼，然後編輯產生的範本以使用動態參數。
- 產生的範本中的目標會保持與原始事件匯流排中指定的完全相同。如果您在使用範本在其他地區建立堆疊之前未適當地編輯範本，這可能會導致跨區域問題。

  此外，產生的範本不會自動建立下游目標。

Amazon EventBridge 目標

目標是當事件符合為規則定義的事件模式時，會將事件 EventBridge 傳送至的資源或端點。規則會處理事件資料，並將相關資訊傳送至目標。若要將事件資料傳遞至目標， EventBridge 需要存取目標資源的權限。您最多可以為每個規則定義五個目標。

當您將目標新增至規則且該規則不久會運行時，可能不會立即調用任何新的目標或更新的目標。允許一小段時間來讓變更生效。

下列影片涵蓋目標的基本概念：什麼是目標

EventBridge 控制台中可用的目標

您可以在 EventBridge 主控台中為事件設定下列目標：

- API 目標
- API Gateway
- AWS AppSync;
- 批次任務佇列
- CloudWatch 記錄群組
- CodeBuild 項目
- CodePipeline
- Amazon EBS `CreateSnapshot` API 呼叫
- EC2 Image Builder
- EC2 `RebootInstances` API 呼叫
- EC2 `StopInstances` API 呼叫
- EC2 `TerminateInstances` API 呼叫
- ECS 任務
- 不同帳戶或地區中的事件匯流排
- 相同帳戶和地區中的事件匯流排
- Firehose 交付串流
- Glue 工作流程
- Indent Manager 回應計劃

- Inspector 評估範本
- Kinesis 串流
- Lambda 函數 (ASYNC)
- Amazon Redshift 叢集資料 API 查詢
- Amazon Redshift Serverless 工作群組資料 API 查詢
- SageMaker 管道
- Amazon SNS 主題

  EventBridge 不支援 Amazon SNS FIFO (先進先出) 主題。
- Amazon SQS 佇列
- Step Functions 狀態機器 (ASYNC)
- Systems Manager Automation
- Systems Manager OpsItem
- Systems Manager Run Command

目標參數

有些目標不會將事件裝載中的資訊傳送至目標，而是將事件視為叫用特定 API 的觸發器。 EventBridge 使用 Target 參數來決定該目標會發生什麼情況。這些索引標籤包括以下項目：

- API 目的地 (傳送至 API 目的地的資料必須與 API 的結構相符。您必須使用 `InputTransformer` 物件來確保資料結構正確。如果您想要包含原始事件裝載，請在 `InputTransformer` 中進行參考。)
- API Gateway (傳送至 API Gateway 的資料必須與 API 的結構相符。您必須使用 `InputTransformer` 物件來確保資料結構正確。如果您想要包含原始事件裝載，請在 `InputTransformer` 中進行參考。)
- Amazon EC2 Image Builder
- `RedshiftDataParameters` (Amazon Redshift 資料 API 叢集)
- `SageMakerPipelineParameters` (Amazon SageMaker 運行時模型構建管道)

> **Note**
>
> EventBridge 不支持所有 JSON 路徑語法，並在運行時對其進行評估。支援的語法包括：
>
> - 點符號 (例如 `$.detail`)

- 破折號
- 底線
- 英數字元
- 陣列索引
- 萬用字元 (*)

動態路徑參數

某些目標參數支援選用的動態 JSON 路徑語法。此語法可讓您指定 JSON 路徑，而非靜態值 (例如 `$.detail.state`)。整個值必須是 JSON 路徑，而不僅僅是其中的一部分。例如，`RedshiftParameters.Sql` 可以是 `$.detail.state` 但不能是 `"SELECT * FROM $.detail.state"`。這些路徑會在執行期以指定路徑中的事件裝載本身的資料動態取代。動態路徑參數無法參考輸入轉換所產生的新值或轉換值。動態參數 JSON 路徑支援的語法與轉換輸入時相同。如需更多資訊，請參閱???

動態語法可以用於所有字符串，這些參數的非枚舉字段：

- `EcsParameters`
- `HttpParameters` (`HeaderParameters` 按鍵除外)
- `RedshiftDataParameters`
- `SageMakerPipelineParameters`

許可

若要對您擁有的資源進行 API 呼叫， EventBridge 需要適當的權限。對於 AWS Lambda 和 Amazon SNS 資源，請 EventBridge 使用以資源為基礎的政策。對於 EC2 執行個體、Kinesis 資料串流和 Step Functions 數狀態機器， EventBridge 會使用您在中的`RoleARN`參數中`PutTargets`指定的 IAM 角色。您可以調用具有已設定 IAM 授權的 API Gateway 端點，但如果您尚未設定授權，則該角色為選用角色。如需詳細資訊，請參閱 Amazon EventBridge 和 AWS Identity and Access Management。

如果另一個帳戶位於同一地區，並已授予您許可，您可以將事件傳送至該帳戶。如需詳細資訊，請參閱在 AWS 帳戶之間傳送和接收 Amazon EventBridge 事件。

如果您的目標已加密，則必須在 KMS 金鑰政策中包含下列區段。

```
{
    "Sid": "Allow EventBridge to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": "events.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

EventBridge 目標細節

AWS Batch 工作佇列

某些參數 AWS Batch `submitJob`可以通過配置BatchParameters。

其他可以在事件裝載中進行指定。如果事件裝載 (傳遞或通過 InputTransformers) 包含下列索引鍵，則會將它們對應至`submitJob`要求參數：

- `ContainerOverrides: containerOverrides`

  > **Note**
  >
  > 這僅包括命令、環境、記憶體和 vcpus

- `DependsOn: dependsOn`

  > **Note**
  >
  > 這僅包括 jobId

- `Parameters: parameters`

CloudWatch 記錄檔群組

如果您未InputTransformer搭配 CloudWatch Logs 目標使用，則會使用事件裝載作為記錄訊息，而事件來源作為時間戳記使用。如果您使用的是 InputTransformer，範本必須是：

```
{"timestamp":<timestamp>,"message":<message>}
```

EventBridge 批次傳送至記錄串流的項目；因此，視流量而定， EventBridge 可能會將單一或多個事件傳遞至記錄串流。

CodeBuild 項目

如果您使InputTransformers用將輸入事件塑造成 Target 以符合結 CodeBuild StartBuildRequest構，則參數會對應 1 對 1 並傳遞至。`codeBuild.StartBuild`

Amazon ECS 任務

如果您使InputTransformers用將輸入事件塑造為 Target 以符合 Amazon ECS RunTask TaskOverride結構，則參數將會對應 1 對 1 並傳遞至。`ecs.RunTask`

Incident Manager 回應計劃

如果相符的事件來自 CloudWatch [警示]，警示狀態變更詳細資料會填入事件管理員 StartIncidentRequest 呼叫的觸發器詳細資料中。

設定目標

瞭解如何設定 EventBridge 目標的設定。

目標：

- API 目的地
- Amazon API 網關的亞馬遜 EventBridge 目標
- AWS AppSync Amazon 的目標 EventBridge
- HTTP 端點目標的連線
- 在 AWS 帳戶之間傳送和接收 Amazon EventBridge 事件
- 在 AWS 區域之間傳送和接收 Amazon EventBridge 事件
- 在同一帳戶和區域的事件匯流排之間傳送和接收 Amazon EventBridge 事件

API 目的地

Amazon EventBridge API 目的地是可以呼叫做為規則目標的 HTTP 端點，類似於將 AWS 服務或資源作為目標叫用的方式。使用 API 目的地，您可以使用 API 呼叫，在 AWS 服務、整合式軟體即服務 (SaaS) 應用程式以及外部的應用程式路 AWS 由事件。當您指定 API 目的地作為規則的目標時，會針對符合規則中指定之事件模式的任何事件 EventBridge 叫用 HTTP 端點，然後將事件資訊與要求一起傳送。使用時 EventBridge，您可以針對要求使用除「連線」和「追蹤」以外的任何 HTTP 方法。要使用的最常見 HTTP 方法是 PUT 和 POST。您也可以使用輸入轉換器，將事件自訂為特定 HTTP 端點參數的參數。如需詳細資訊，請參閱 Amazon EventBridge 輸入轉換。

API 目的地不支援私有目的地，例如介面 VPC 端點。如需詳細資訊，請參閱 ???。

> **Important**
>
> EventBridge 對 API 目標端點的要求必須具有 5 秒的用戶端執行逾時上限。如果目標端點需要超過 5 秒的 EventBridge 時間來回應，請求逾時。 EventBridge 重試將要求逾時，直到重試原則上設定的上限為止。預設情況下，最大值為 24 小時和 185 次。在重試次數上限之後，如果您有無效字母佇列，則會將事件傳送至無效字母佇列。否則，會捨棄該事件。

下列影片示範 API 目的地的使用：使用 API 目的地

在本主題中：

建立 API 目的地

每個 API 目的地都需要連線。連線會定義用於向 API 目的地端點授權的授權類型和憑證。您可以選擇現有連線，或在建立 API 目的地的同時建立連線。如需更多資訊，請參閱???

使用 EventBridge 主控台建立 API 目的地

1. AWS 使用具有管理 EventBridge 和開啟EventBridge主控台權限的帳戶登入。
2. 在左側導覽窗格中選擇 API 目的地。
3. 向下捲動至 API 目的地表格，然後選擇建立 API 目的地。
4. 在建立 API 目的地頁面上，輸入 API 目的地的名稱。您最多可使用 64 個大寫或小寫字母、數字、點 (.)、破折號 (-) 或底線 (_) 字元。

   在當前區域中，此名稱對於您的帳戶必須是唯一的。
5. 輸入 API 目的地的描述。
6. 輸入 API 目的地的 API 目的地端點。API 目的地端點是事件的 HTTP 調用端點目標。您在用於此 API 目的地的連線中包含的授權資訊會用來對此端點進行授權。URL 必須使用 HTTPS。
7. 輸入用於連線至 API 目的地端點的 HTTP 方法。
8. (選用) 在每秒調用速率限制欄位中，輸入每秒要傳送至 API 目的地端點的調用數目上限。

   您設定的速率限制可能會影響 EventBridge 傳送事件的方式。如需詳細資訊，請參閱 調用率如何影響事件交付。
9. 針對連線，執行下列其中一項作業：
   - 選擇使用現有連線，然後選取要用於此 API 目的地的連線。
   - 選擇建立新連線，然後輸入要建立之連線的詳細資訊。如需詳細資訊，請參閱連線。
10. 選擇建立。

建立將事件傳送至 API 目的地的規則

建立 API 目的地後，您可以選取它作為規則的目標。若要使用 API 目的地做為目標，您必須提供具有正確許可的 IAM 角色。如需更多資訊，請參閱???

選取 API 目標作為目標是建立規則的一部分。

建立使用主控台將事件傳送至 API 目的地的規則

1. 然後依照 ??? 程序中的步驟進行操作。

2. 在???步驟中，當系統提示您選擇 API 目標作為目標類型時：

   a. 選取 EventBridge API 目的地。

   b. 執行以下任意一項：

      - 選擇使用現有的 API 目的地，然後選擇現有的 API 目的地
      - 選擇建立新的 API 目的地，並指定必要的設定來定義新的 API 目的地。

        如需指定所需設定的詳細資訊，請參閱???。

   c. (選擇性)：若要指定事件的標頭參數，請在「標頭參數」下選擇「新增標頭參數」。

      接下來，指定標頭參數的鍵和值。

   d. (選擇性)：若要指定事件的查詢字串參數，請在 [查詢字串參數] 下選擇 [新增查詢字串參數]。

      接下來，指定查詢字串參數的索引鍵和值。

3. 按照程序步驟完成建立規則。

API 目的地的服務連結角色

當您為 API 目標建立連線時，名AWS ServiceRoleForAmazonEventBridgeApiDestinations為的服務連結角色會新增至您的帳戶。 EventBridge 使用服務連結的角色在 Secret 管理員中建立和儲存密碼。若要將必要的權限授與服務連結角色，請將AmazonEventBridgeApiDestinationsServiceRolePolicy原則EventBridge附加至角色。此政策會限制僅授予該角色與連線機密互動所需的許可。不包含其他許可，且該角色僅能與您帳戶中的連線進行互動以管理機密。

下面的政策為 `AmazonEventBridgeApiDestinationsServiceRolePolicy`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:DeleteSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue"
            ],
```

```
            "Resource": "arn:aws:secretsmanager:*:*:secret:events!connection/*"
        }
    ]
}
```

如需有關服務連結角色的詳細資訊，請參閱 IAM 文件中的使用服務連結角色。

下列 AWS 地區支援AmazonEventBridgeApiDestinationsServiceRolePolicy服務連結角色：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太區域 (香港)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 歐洲 (米蘭)
- 南美洲 (聖保羅)
- 中國 (寧夏)
- 中國 (北京)

API 目的地之請求中的標頭

以下部分詳細說明如何 EventBridge 處理 API 目的地請求中的 HTTP 標頭。

API 目的地請求中包含的標頭

除了針對用於 API 目的地之連線定義的授權標頭之外，每個要求中都 EventBridge 包含下列標頭。

標頭鍵	標頭值
使用者代理程式	Amazon/EventBridgeApiDestinations
內容類型	如果未指定自訂內容類型值，請將下列預設值 EventBridge 包含為內容類型： application/json; charset=utf-8
範圍	bytes=0-1048575
接受編碼	gzip,deflate
連線	關閉
內容長度	實體標頭，是指傳送給收件者的實體主體大小 (以位元組為單位)。
主機	請求標頭，指定要傳送請求之伺服器的主機和連接埠號碼。

在 API 目的地之請求中無法覆寫的標頭

EventBridge 不允許您覆寫下列標頭：

- 使用者代理程式
- 範圍

標頭 EventBridge 會從要求移除 API 目的地

EventBridge 移除所有 API 目標要求的下列標頭：

- A-IM
- Accept-Charset
- Accept-Datetime
- 接受編碼
- 快取控制
- 連線
- Content-Encoding
- 內容長度
- Content-MD5
- 日期
- Expect
- Forwarded
- 從
- 主機
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Origin
- Pragma
- Proxy-Authorization
- 範圍
- Referer
- TE
- 預告片
- Transfer-Encoding

- 使用者代理程式
- 升級
- Via
- 警告

API 目的地錯誤代碼

嘗 EventBridge 試將事件傳送到 API 目的地並發生錯誤時，請 EventBridge執行以下操作：

- 會重試與錯誤碼 409、429 和 5xx 相關聯的事件。
- 與錯誤代碼 1xx、2xx、3xx 和 4xx (不包括 429) 相關聯的事件不會重試。

EventBridge API 目的地讀取標準 HTTP 回應標頭，`Retry-After`以瞭解在提出後續請求之前要等待多長時間。 EventBridge 會在定義的重試原則和`Retry-After`標頭之間選取較保守的值。如果`Retry-After`值為負數，則 EventBridge 會停止重試該事件的傳送。

調用率如何影響事件交付

如果您將每秒的調用率設定為遠低於所產生之調用數目的值，則事件可能無法在 24 小時的重試時間內交付事件。例如，如果您將調用速率設定為每秒 10 次調用，但每秒產生數千個事件，則您很快就會有待交付超過 24 小時的待處理事件。若要確保沒有遺失任何事件，請設定無效字母佇列來傳送調用失敗的事件，以便您稍後可以處理這些事件。如需詳細資訊，請參閱 事件重試政策和使用無效字母佇列。

將 CloudEvents 事件傳送至 API 目的地

CloudEvents 是事件格式化的供應商中立規格，其目標是在服務、平台和系統之間提供互通性。您可以使用 EventBridge 在 AWS 服務事件傳送至目標 (例如 API 目的地) CloudEvents 之前將服務事件轉換為。

> **Note**
>
> 下列程序說明如何將來源事件轉換成結構 CloudEvents模式。在 CloudEvents 規格中，結構模式消息是將整個事件（屬性和數據）編碼到事件的有效負載中的消息。

如需有關 CloudEvents 規格的詳細資訊，請參閱雲端文字 .io。

使用控制台將 AWS 事件轉換為 CloudEvents 格式

若要在傳遞至目標之前將事件轉換為 CloudEvents 格式，請先建立事件匯流排規則。在定義規則的過程中，您可以使用輸入 EventBridge 轉換器在傳送至您指定的目標之前發生轉換事件。

1. 然後依照 ??? 程序中的步驟進行操作。
2. 在???步驟中，當系統提示您選擇 API 目標作為目標類型時：

   a. 選取 EventBridge API 目的地。

   b. 執行以下任意一項：

      - 選擇使用現有的 API 目的地，然後選擇現有的 API 目的地
      - 選擇建立新的 API 目的地，並指定必要的設定來定義新的 API 目的地。

        如需指定所需設定的詳細資訊，請參閱???。

   c. 為事件指定必要的內容類型標頭參數 CloudEvents ：

      - 在「表頭參數」下選擇「新增標頭參數
      - 對於索引鍵，請指定`Content-Type`。

        對於值，請指定`application/cloudevents+json; charset=UTF-8`。
3. 指定目標的執行角色。
4. 定義輸入轉換器，將來源事件資料轉換為以下 CloudEvents 格式：

   a. 在其他設定下，對於設定目標輸入，選擇輸入變壓器。

      然後選擇設定輸入變壓器。

   b. 在「目標輸入變壓器」下，指定「輸入路徑」。

      在下面的輸入路徑中，region 屬性是 CloudEvents 格式的自定義擴展屬性。因此，它不是必需的 CloudEvents 遵守規範。

      CloudEvents 允許您使用和創建未在核心規格中定義的擴展屬性。如需詳細資訊，包括已知擴充功能屬性清單，請參閱上的CloudEvents 規格文件中的CloudEvents 擴充功能屬性 GitHub。

```
{
  "detail": "$.detail",
  "detail-type": "$.detail-type",
```

```
  "id": "$.id",
  "region": "$.region",
  "source": "$.source",
  "time": "$.time"
}
```

c. 在「範本」中，輸入要將來源事件資料轉換為 CloudEvents 格式的範本。

在下面的模板中，不`region`是嚴格要求的，因為輸入路徑中的`region`屬性是 CloudEvents 規格的擴展屬性。

```
{
  "specversion":"1.0",
  "id":<id>,
  "source":<source>,
  "type":<detail-type>,
  "time":<time>,
  "region":<region>,
  "data":<detail>
}
```

5. 按照程序步驟完成建立規則。

API 目的地合作夥伴

使用下列 AWS 合作夥伴提供的資訊，為其服務或應用程式設定 API 目的地和連線。

合口

API 目的地調用端點網址：

通常格式如下：

`https://random-id.region.aws.confluent.cloud:443/kafka/v3/clusters/cluster-id/topics/topic-name/records`

如需詳細資訊，請參閱 Confluent 文件中的尋找 REST 端點位址和叢集識別碼。

支援的授權類型：

基本

需要其他授權參數：

不適用

匯合文檔：

產生記錄

阿帕奇卡夫卡的匯合 REST 代理

常用的 API 操作：

POST

其他資訊：

若要將事件資料轉換為端點可處理的訊息，請建立目標輸入轉換器。

- 要在不指定 Kafka 分區密鑰的情況下生成記錄，請為輸入轉換器使用以下模板。不需要輸入路徑。

```
{
  "value":{
      "type":"JSON",
      "data":aws.events.event.json
  },
}
```

- 若要使用事件資料欄位做為 Kafka 分割金鑰來產生記錄，請遵循下面的輸入路徑和範本範例。此範例會定義orderId欄位的輸入路徑，然後將該欄位指定為磁碟分割索引鍵。

  首先，定義事件資料欄位的輸入路徑：

```
{
  "orderId":"$.detail.orderId"
}
```

  然後，使用輸入轉換器範本將資料欄位指定為磁碟分割索引鍵：

```
{
  "value":{
    "type":"JSON",
    "data":aws.events.event.json
  },
  "key":{
```

```
    "data":"<orderId>",
    "type":"STRING"
  }
}
```

Coralogix

API 目的地調用端點網址

如需完整的端點清單，請參閱 Coralogix API 參考。

支援的授權類型

API 金鑰

需要其他授權參數

標頭 `"x-amz-event-bridge-access-key"`，值為 Coralogix API 金鑰

Coralogix 文件

Amazon EventBridge 認證

常用的 API 操作

美國：https://ingress.coralogix.us/aws/event-bridge

新加坡：https://ingress.coralogixsg.com/aws/event-bridge

愛爾蘭：https://ingress.coralogix.com/aws/event-bridge

斯德哥爾摩：https://ingress.eu2.coralogix.com/aws/event-bridge

印度：https://ingress.coralogix.in/aws/event-bridge

其他資訊

這些事件會儲存為具有 `applicationName=[AWS Account]` 和 `subsystemName=[event.source]` 的記錄項目。

Datadog

API 目的地調用端點網址

如需完整的端點清單，請參閱 Datadog API 參考。

支援的授權類型

API 金鑰

需要其他授權參數

無

Datadog 文件

身分驗證

常用的 API 操作

POST https://api.datadoghq.com/api/v1/events

POST https://http-intake.logs.datadoghq.com/v1/input

其他資訊

端點 URL 會根據您的 Datadog 組織的位置而有所不同。如需您的組織的正確 URL，請參閱文件。

Freshworks

API 目的地調用端點網址

如需端點的清單，請參閱 https://developers.freshworks.com/documentation/。

支援的授權類型

基本，API 金輪

需要其他授權參數

不適用

Freshworks 文件

身分驗證

常用的 API 操作

https://developers.freshdesk.com/api/#create_ticket

https://developers.freshdesk.com/api/#update_ticket

https://developer.freshsales.io/api/#create_lead

https://developer.freshsales.io/api/#update_lead

其他資訊

無

MongoDB

API 目的地調用端點網址

https://data.mongodb-api.com/app/*App ID*/endpoint/

支援的授權類型

API 金鑰

電子郵件/密碼

自訂 JWT 身分驗證

需要其他授權參數

無

MongoDB 文件

Atlas 資料 API

端點

自訂 HTTPS 端點

身分驗證

常用的 API 操作

無

其他資訊

無

New Relic

API 目的地調用端點網址

如需詳細資訊，請參閱我們的歐盟和美國地區資料中心。

事件

美國: https://insights-collector.newrelic.com/v1/accounts/*YOUR_NEW_RELIC_ACCOUNT_ID*/events

歐盟： https://insights-collector.eu01.nr-data.net/v1/accounts/*YOUR_NEW_RELIC_ACCOUNT_ID*/events

指標

美國： https://metric-api.newrelic.com/metric/v1

歐盟： https://metric-api.eu.newrelic.com/metric/v1

日誌

美國： https://log-api.newrelic.com/log/v1

歐盟： https://log-api.eu.newrelic.com/log/v1

追蹤

美國： https://trace-api.newrelic.com/trace/v1

歐盟： https://trace-api.eu.newrelic.com/trace/v1

支援的授權類型

API 金鑰

New Relic 文件

指標 API

事件 API

日誌 API

追蹤 API

常用的 API 操作

指標 API

事件 API

日誌 API

追蹤 API

其他資訊

指標 API 限制

事件 API 限制

日誌 API 限制

追蹤 API 限制

Operata

API 目的地調用端點網址：

https://api.operata.io/v2/aws/events/contact-record

支援的授權類型：

基本

需要其他授權參數：

無

Operata 文件：

如何建立、查看、變更和撤銷 API 權杖？

使用 Amazon EventBridge 調度管道進行操作 AWS 集成

常用的 API 操作：

```
POST https://api.operata.io/v2/aws/events/contact-record
```

其他資訊：

`username` 為 Operata Group ID，密碼為您的 API 權杖。

Salesforce

API 目的地調用端點網址

插件-HTTPS://myDomainName.我的.salesforce.com/服務/數據/版本號/sobject *//* *SobjectEndpoint*

自訂平台事件：*//myDomainName.my.salesforce.com ##/##/#### customPlatformEndpoint*

如需完整的端點清單，請參閱 Salesforce API 參考

支援的授權類型

OAuth 用戶端憑證

傳回 401 或 407 回應時，OAUTH 權杖會被重新整理。

需要其他授權參數

Salesforce 連線的應用程式用戶端 ID 和用戶端機密。

下列其中一個授權端點：

- 生產- HTTPS://*MyDomainName*. 我的. /服務/Oauth2/代幣
- 沒有增強網域的沙箱 — HTTPS://*MyDomainName- SandboxName*. 我的服務
- 具有增強網域的沙箱：*MyDomainName SandboxName*

以下鍵/值對：

索引鍵	值
grant_type	client_credentials

Salesforce 文件

REST API 開發人員指南

常用的 API 操作

使用物件中繼資料

使用記錄

其他資訊

如需說明如何使用 EventBridge 主控台建立連線、API 目的地Salesforce，以及將資訊路由到的規則的教學課程Salesforce，請參閱???。

Slack

API 目的地調用端點網址

如需端點和其他資源的清單，請參閱使用 Slack Web API

支援的授權類型

OAuth 2.0

傳回 401 或 407 回應時，OAUTH 權杖會被重新整理。

當您建立 Slack 應用程序並將其安裝到工作區時，將代表您會建立 OAuth 承載權杖，以用於透過 API 目的地連線的驗證呼叫。

需要其他授權參數

不適用

Slack 文件

基本的應用程式設定

使用 OAuth 進行安裝

擷取訊息

傳送訊息

使用傳入 Webhook 傳送訊息

常用的 API 操作

https://slack.com/api/chat.postMessage

其他資訊

配置 EventBridge 規則時，有兩種配置可以突出顯示：

- 包括將內容類型定義為“application/json;charset=utf-8”的標題參數。
- 使用輸入轉換器將輸入事件映射到 Slack API 的預期輸出，即確保傳送至 Slack API 的承載具有“通道”和“文字”鍵/值對。

Shopify

API 目的地調用端點網址

如需端點和其他資源和方法的清單，請參閱端點和請求

支援的授權類型

OAuth，API 金鑰

> **Note**
>
> 傳回 401 或 407 回應時，OAUTH 權杖會被重新整理。

需要其他授權參數

不適用

Shopify 文件

驗證與授權概觀

常用的 API 操作

POST - /admin/api/2022-01/products.json

GET - admin/api/2022-01/products/{product_id}.json

PUT - admin/api/2022-01/products/{product_id}.json

DELETE - admin/api/2022-01/products/{product_id}.json

其他資訊

建立應用程式

Amazon EventBridge 網絡掛鉤交付

在 Shopify 管理員中存取自訂應用程式的權杖

產品

Shopify 管理員 API

Splunk

API 目的地調用端點網址

https://*SPLUNK_HEC_ENDPOINT*:*optional_port*/services/collector/raw

支援的授權類型

基本，API 金輪

需要其他授權參數

無

Splunk 文件

對於這兩種授權類型，您都需要一個 HEC 權杖 ID。如需詳細資訊，請參閱在 Splunk Web 中設定和使用 HTTP 事件收集器。

常用的 API 操作

POST https://*SPLUNK_HEC_ENDPOINT*:*optional_port*/services/collector/raw

其他資訊

API 金鑰 — 設定的端點時 EventBridge，API 金鑰名稱為「授權」，而值是 Splunk HEC 權杖識別碼。

基本 (使用者名稱/密碼) — 設定的端點時 EventBridge，使用者名稱為「Splunk」，密碼是 Splunk HEC 權杖識別碼。

Sumo Logic

API 目的地調用端點網址

每個使用者的 HTTP 日誌和指標來源端點 URL 將不同。如需詳細資訊，請參閱 HTTP 日誌和指標來源。

支援的授權類型

Sumo Logic 不需要對其 HTTP 源進行身分驗證，因為 URL 中有一個唯一金輪。因此，您應該確保將 URL 視為機密。

設定 EventBridge API 目的地時，需要授權類型。為了滿足此要求，請選取 API 金輪並為其指定“虛擬金輪”的金輪名稱和“虛擬值”的金輪值。

需要其他授權參數

不適用

Sumo Logic 文件

Sumo Logic已經構建了託管源來收集來自許多 AWS 服務的日誌和指標，您可以使用其網站上的信息與這些來源一起使用。如需詳細資訊，請參閱 Amazon Web Services。

如果您要從應用程式產生自訂事件，並希望將其Sumo Logic作為記錄檔或指標傳送至，請使用 EventBridge API 目的地和 Sumo Logic HTTP 記錄和指標來源端點。

- 若要註冊並建立免費 Sumo Logic 執行個體，請參閱立即開始免費試用。
- 如需有關使用 Sumo Logic 的詳細資訊，請參閱 HTTP 日誌和指標來源。

常用的 API 操作

POST https://endpoint4.collection.us2.sumologic.com/receiver/v1/http/*UNIQUE_ID_PER_COLLECTOR*

其他資訊

無

TriggerMesh

API 目的地調用端點網址

使用 HTTP 事件來源主題中的資訊來制定端點 URL。端點 URL 包含事件來源名稱和使用者命名空間，格式如下：

https://*source-name*.*user-namespace*.cloud.triggermesh.io

在針對端點的請求中包含基本的授權參數。

支援的授權類型

基本

需要其他授權參數

無

TriggerMesh 文件

HTTP 的事件來源

常用的 API 操作

不適用

其他資訊

無

Zendesk

API 目的地調用端點網址

https://developer.zendesk.com/rest_api/docs/support/tickets

支援的授權類型

基本，API 金輪

需要其他授權參數

無

Zendesk 文件

安全性與驗證

常用的 API 操作

POST https://*your_Zendesk_subdomain*/api/v2/tickets

其他資訊

API 請求 EventBridge 會根據您的 Zendesk API 限制進行計數。如需有關您的方案之 Zendesk 限制的資訊，請參閱用量限制。

為了更好地保護您的帳戶和資料，我們建議您使用 API 金輪而不是基本的登錄憑證身分驗證。

Amazon API 網關的亞馬遜 EventBridge 目標

您可以使用 Amazon API Gateway 來建立、發佈、維護與監控 API。Amazon EventBridge 支援將事件傳送到 API Gateway 端點。當您指定 API Gateway 端點做為目標時，傳送至目標的每個事件都會映射至傳送至端點的請求。

> **Important**
>
> EventBridge 支援使用 API Gateway 邊緣最佳化和區域端點做為目標。目前不支援私人端點。若要進一步了解叢集端點，請參閱 https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-endpoint-types.html。

您可以針對下列使用案例使用 API Gateway 目標：

- 根據 AWS 或第三方事件叫用在 API Gateway 中託管的客戶指定 API。
- 按照排程定期調用端點。

EventBridge JSON 事件資訊會做為 HTTP 要求的主體傳送至您的端點。您可以在目標 `HttpParameters` 欄位中指定其他請求屬性，如下所示：

- `PathParameterValues` 列出了與端點 ARN 中的任何路徑變量順序對應的值，例如 `"arn:aws:execute-api:us-east-1:112233445566:myapi/dev/POST/pets/*/*"`。
- `QueryStringParameters`表示 EventBridge 附加到叫用端點的查詢字串參數。
- `HeaderParameters` 定義了要新增至請求的 HTTP 標頭。

> **Note**
>
> 基於安全性考量，不允許使用下列 HTTP 標頭金鑰：
>
> - 任何前綴為 `X-Amz` 或 `X-Amzn`
> - `Authorization`
> - `Connection`
> - `Content-Encoding`
> - `Content-Length`
> - `Host`
> - `Max-Forwards`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`

- Upgrade
- Via
- WWW-Authenticate
- X-Forwarded-For

動態參數

調用 API Gateway 目標時，您可以動態地將資料新增至傳送至目標的事件。如需詳細資訊，請參閱 the section called “目標參數”。

調用指標

與所有目標一樣， EventBridge 重試一些失敗的調用。對於 API Gateway， EventBridge 重試使用 5xx 或 429 HTTP 狀態碼傳送的回應，最長可達 24 小時，並具有指數退回和抖動。之後，在 Amazon 中 EventBridge 發布`FailedInvocations`指標 CloudWatch。 EventBridge 不會重試其他 4xx HTTP 錯誤。

逾時

EventBridge 規則 API Gateway 要求的用戶端執行逾時上限必須為 5 秒。如果 API Gateway 需要超過 5 秒的 EventBridge 時間來回應，請求逾時，然後重試。

EventBridge 管道 API Gateway 請求的逾時上限為 29 秒，即 API Gateway 上限。

AWS AppSync Amazon 的目標 EventBridge

AWS AppSync 使開發人員能夠使用安全、無伺服器和高效能 GraphQL 和 Pub/Sub API，將其應用程式和服務與資料和事件連接起來。您可以使用 GraphQL 突變 AWS AppSync，將即時資料更新發佈到您的應用程式。 EventBridge 支援針對相符事件呼叫有效的 GraphQL 突變作業。當您指定 AWS AppSync API 突變作為目標時，會透過突變作業來 AWS AppSync 處理事件，然後觸發連結至突變的訂閱。

> **Note**
>
> EventBridge 支援 AWS AppSync 公用的 GraphQL API。 EventBridge 目前不支援 AWS AppSync 私有 API。

您可以針對下列使用案例使用 AWS AppSync GraphQL API 目標：

- 將事件資料推送、轉換和儲存到您設定的資料來源中。
- 將即時通知傳送至連線的應用程式用戶端。

> **Note**
>
> AWS AppSync 目標僅支援使用AWS_IAM授權類型呼叫 AWS AppSync GraphQL API。

如需 AWS AppSync GraphQL API 的詳細資訊，請參閱AWS AppSync 開發人員指南中的 GraphQL 和 AWS AppSync 架構。

使用控制台指 AWS AppSync 定 EventBridge 規則的目標

1. 建立或編輯此規則。
2. 在目標下，依序選擇 AWS 服務、AWS AppSync，來指定目標。
3. 指定要剖析和執行的突變作業，以及選取集。
   - 選擇該 AWS AppSync API，然後選擇要調用的 GraphQL API 突變。
   - 在設定參數和選取集下，選擇使用鍵-值對映或輸入轉換器來建立選取集。

     Key-value mapping

     若要使用鍵-值對映來建立選取集：
     - 指定 API 參數的變數。每個變數都可以是靜態值，也可以是事件承載的動態 JSON 路徑表達式。
     - 在選取集下，選擇您要在回應中包含的變數。

     Input transformer

     若要使用輸入轉換器建立選取集：
     - 指定定義要使用之變數的輸入路徑。
     - 指定輸入範本，以定義和格式化您要傳遞至目標的資訊。

     如需詳細資訊，請參閱 ???。
4. 對於執行角色，選擇是否要新建角色，或使用現有的角色。
5. 完成規則的建立或編輯。

示例：Amazon 的 AWS AppSync 目標 EventBridge

在下列範例中，我們將逐步介紹如何指定 EventBridge 規則的 AWS AppSync 目標，包括定義輸入轉換以設定要傳遞之事件格式的輸入轉換。

假設您有一個 AWS AppSync GraphQL API`Ec2EventAPI`，由下列結構描述定義：

```
type Event {
    id: ID!
    statusCode: String
    instanceId: String
}

type Mutation {
    pushEvent(id: ID!, statusCode: String!, instanceId: String): Event
}

type Query {
    listEvents: [Event]
}

type Subscription {
    subscribeToEvent(id: ID, statusCode: String, instanceId: String): Event
        @aws_subscribe(mutations: ["pushEvent"])
}
```

使用此 API 的應用程式用戶端可以訂閱 `subscribeToEvent`，該訂閱是由 `pushEvent` 突變所觸發。

您可以建立具有透過`pushEvent`變異將事件傳送至 AppSync API 的目標的 EventBridge 規則。調用突變時，任何訂閱的用戶端都會收到該事件。

若要將此 API 指定為 EventBridge 規則的目標，您可以執行下列動作：

1. 將規則目標的 Amazon Resource Name (ARN) 設為 `Ec2EventAPI` API 的 GraphQL 端點 ARN。
2. 將突變 GraphQL 作業指定為目標參數：

```
mutation CreatePushEvent($id: ID!, $statusCode: String, $instanceId: String) {
  pushEvent(id: $input, statusCode: $statusCode, instanceId: $instanceId) {
    id
    statusCode
    instanceId
  }
```

```
}
```

突變選取集必須包含您希望在 GraphQL 訂閱中訂閱的所有欄位。

3. 設定輸入轉換器，以指定如何在作業中使用相符事件中的資料。

   假設您選取了“EC2 Instance Launch Successful”範例事件：

```
{
  "version": "0",
  "id": "3e3c153a-8339-4e30-8c35-687ebef853fe",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2015-11-11T21:31:47Z",
  "region": "us-east-1",
  "resources": ["arn:aws:autoscaling:us-
east-1:123456789012:autoScalingGroup:eb56d16b-bbf0-401d-b893-
d5978ed4a025:autoScalingGroupName/sampleLuanchSucASG", "arn:aws:ec2:us-
east-1:123456789012:instance/i-b188560f"],
  "detail": {
    "StatusCode": "InProgress",
    "AutoScalingGroupName": "sampleLuanchSucASG",
    "ActivityId": "9cabb81f-42de-417d-8aa7-ce16bf026590",
    "Details": {
      "Availability Zone": "us-east-1b",
      "Subnet ID": "subnet-95bfcebe"
    },
    "RequestId": "9cabb81f-42de-417d-8aa7-ce16bf026590",
    "EndTime": "2015-11-11T21:31:47.208Z",
    "EC2InstanceId": "i-b188560f",
    "StartTime": "2015-11-11T21:31:13.671Z",
    "Cause": "At 2015-11-11T21:31:10Z a user request created an AutoScalingGroup
 changing the desired capacity from 0 to 1.  At 2015-11-11T21:31:11Z an instance was
 started in response to a difference between desired and actual capacity, increasing
 the capacity from 0 to 1."
  }
}
```

您可以使用目標輸入轉換器的輸入路徑，定義要在範本中使用的下列變數：

```
{
  "id": "$.id",
```

```
  "statusCode": "$.detail.StatusCode",
  "EC2InstanceId": "$.detail.EC2InstanceId"
}
```

撰寫輸入轉換器範本，以定義 EventBridge 傳遞至變 AWS AppSync 異作業的變數。此範本必須評估為 JSON。考量到輸入路徑，您可以撰寫以下範本：

```
{
  "id": <id>,
  "statusCode": <statusCode>,
  "instanceId": <EC2InstanceId>
}
```

HTTP 端點目標的連線

連線會定義用於連線 EventBridge 至指定 HTTP 端點的授權方法和認證。當您配置授權設置並創建連接時，它會在中創建一個密碼 AWS Secrets Manager 以安全地存儲授權信息。您也可以針對 HTTP 端點目標，新增要包含在連線中的其他參數。

使用連線：

- API 目的地

  當您建立 API 目的地時，您可以指定要用於該目的地的連線。您可以從帳戶中選擇現有的連線，或在建立 API 目的地時建立連線。

連線的授權方法

EventBridge 連線支援下列授權方法：

- 基本
- API 金鑰

  對於基本和 API 金鑰授權，請為您 EventBridge 填入所需的授權標頭。
- OAuth

  對於 OAuth 授權， EventBridge 還將您的客戶端 ID 和密鑰交換為訪問令牌，然後安全地管理它。

  傳回 401 或 407 回應時，OAUTH 權杖會被重新整理。

建立連線時，您也可以包含端點授權所需的標頭、主體和查詢參數。如果端點的授權相同，您可以對多個 HTTP 端點使用相同的連線。

當您建立連線並新增授權參數時， EventBridge 會在中建立密碼 AWS Secrets Manager。存儲和存取機密管理員機密的成本包含在使用 API 目的地的費用中。若要進一步了解使用密碼搭配 API 目的地的最佳做法，請參閱使用CloudFormation 者指南AWS::Events::ApiDestination中的。

> **Note**
>
> 若要成功建立或更新連線，您必須使用具有使用 Secrets Manager 許可的帳號。所需的許可包含在 AmazonEventBridgeFullAccess 政策 中。針對在您的帳戶中為連線建立的服務連結角色，也會授予相同的許可。

建立 HTTP 端點目標的連線

使用 EventBridge 主控台建立連線以與 HTTP 端點搭配使用

1. AWS 使用具有管理 EventBridge 和開啟EventBridge 主控台權限的帳戶登入。
2. 在左側導覽窗格中選擇 API 目的地。
3. 向下捲動至 API 目的地表格，然後選擇連線標籤。
4. 選擇建立連線。
5. 在建立連線頁面上，輸入連線的連線名稱。
6. 輸入連線的描述。
7. 針對授權類型，請選取用於授權連線至 HTTP 端點的授權類型，該端點是指定用於使用此連線的 API 目的地。執行以下任意一項：
   - 選擇基本(使用者名稱/密碼)，然後輸入要用來授權 HTTP 端點的使用者名稱和密碼。
   - 選擇 OAuth 用戶端憑證，然後輸入授權端點、HTTP 方法、用戶端 ID 和用戶端機密，以用來授權端點。

     在 OAuth Http 參數下，新增要包含在授權端點之授權的任何其他參數。從下拉式清單中選取參數，然後輸入金鑰和值。若要包括其他參數，請選擇新增參數。

     在調用 Http 參數下，新增要包含在授權請求中的任何其他參數。若要新增參數，請從下拉式清單中選取參數，然後輸入金鑰和值。若要包括其他參數，請選擇新增參數。
   - 選擇 API 金鑰，然後輸入要用於 API 金鑰授權的 API 金鑰名稱和關聯的值。

在調用 Http 參數下，新增要包含在授權請求中的任何其他參數。若要新增參數，請從下拉式清單中選取參數，然後輸入金鑰和值。若要包括其他參數，請選擇新增參數。

8. 選擇建立。

使用 EventBridge 主控台編輯連線

您可以編輯現有的連接。

使用 EventBridge 主控台編輯連線

1. AWS 使用具有管理 EventBridge 和開啟EventBridge 主控台權限的帳戶登入。
2. 在左側導覽窗格中選擇 API 目的地。
3. 向下捲動至 API 目的地表格，然後選擇連線標籤。
4. 在連線表格中，選擇要編輯的連線。
5. 在連線詳細資訊頁面上，選擇編輯。
6. 更新連線的值，然後選擇更新。

使用主控台取消授權連線 EventBridge

取消授權連線時，會移除所有授權參數。移除授權參數會從連線中移除機密，因此您可以重複使用它，而不必建立新的連線。

> **Note**
>
> 您必須更新任何使用取消授權連線的 HTTP 端點，才能使用不同的連線，才能成功將要求傳送至 HTTP 端點。

取消授權連線

1. AWS 使用具有管理 EventBridge 和開啟EventBridge 主控台權限的帳戶登入。
2. 在左側導覽窗格中選擇 API 目的地。
3. 向下捲動至 API 目的地表格，然後選擇連線標籤。
4. 在連線表格中，選擇連線。
5. 在連線詳細資訊頁面上，選擇取消授權。

6. 在取消授權連線？對話方塊中，輸入連線的名稱，然後選擇取消授權。

連線的狀態會變更為取消授權，直到程序完成為止。然後狀態會變更為已取消授權。現在，您可以編輯連線以新增新的授權參數。

在 AWS 帳戶之間傳送和接收 Amazon EventBridge 事件

您可以設 EventBridge 定在 AWS 帳戶中的事件匯流排之間傳送和接收事件。當您設定 EventBridge 為在帳戶之間傳送或接收事件時，您可以指定哪些 AWS 帳戶可以從帳戶中的事件匯流排傳送事件或從事件匯流排接收事件。您也可以允許或拒絕來自與事件匯流排相關聯之特定規則的事件，或來自特定來源的事件。如需詳細資訊，請參閱使用 Amazon EventBridge 資源政策簡化跨帳戶存取

> **Note**
>
> 如果您使用 AWS Organizations，則可以指定組織並授與該組織中所有帳戶的存取權。此外，傳送事件匯流排在傳送事件至其他帳戶時，必須附加 IAM 角色。如需詳細資訊，請參閱 AWS Organizations 使用者指南中的什麼是 AWS Organizations。

> **Note**
>
> 如果您使用「事件管理員」回應計劃作為目標，預設會提供與您帳戶共用的所有回應計劃。

只要目的地區域是支援的跨區域目的地區域，您就可以在所有區域的相同區域內的帳戶之間傳送和接收事件，以及在不同區域的帳戶之間傳送和接收事件。 AWS

設定 EventBridge 為在不同帳戶中傳送事件或從事件匯流排接收事件的步驟如下：

- 在接收者帳戶上，編輯事件匯流排上的權限，以允許指定的 AWS 帳戶、組織或所有 AWS 帳戶將事件傳送到接收者帳戶。
- 在寄件者帳戶上，設定一或多個將接收者帳戶的事件匯流排作為目標的規則。

  如果寄件者帳戶繼承了從 AWS 組織傳送事件的權限，則寄件者帳戶還必須具有 IAM 角色，其政策能夠將事件傳送到接收者帳戶。如果您使用建立 AWS Management Console 以接收者帳戶中事件匯流排為目標的規則，則會自動建立角色。如果使用 AWS CLI，則必須手動建立角色。
- 在接收者帳戶上，設定一個或多個符合來自寄件者帳戶事件的規則。

從一個帳戶傳送到另一個帳戶的事件，會做為自訂事件向傳送帳戶收費。接收帳戶不收費。如需詳細資訊，請參閱 Amazon EventBridge 定價。

接收者帳戶可以設定一個規則，將從寄件者帳戶收到的事件發送到第三個帳戶，但系統不會將這些事件傳送到第三個帳戶。

下列影片涵蓋帳戶之間的路由事件：將事件路由傳送至其他 AWS 帳戶中的匯流排

授予權限以允許來自其他 AWS 帳戶的事件

若要從其他帳戶或組織接受事件，您必須先在帳戶的預設事件匯流排上編輯許可。預設事件匯流排會接受來自 AWS 服務、其他授權 AWS 帳戶和PutEvents呼叫的事件。使用附加至事件匯流排的資源型原則授與或拒絕事件匯流排的許可。在策略中，您可以使用 AWS 帳號 ID 將權限授與其他帳戶，或授與使用 AWS 組織 ID 的組織權限。若要進一步了解事件匯流排許可 (包括範例政策) 的更多資訊，請參閱 Amazon EventBridge 事件匯流排的許可。

> **Note**
>
> EventBridge 現在需要所有新的跨帳戶事件匯流排目標才能新增 IAM 角色。這僅適用於 2023 年 3 月 2 日之後建立的事件匯流排目標。在該日期之前未透過 IAM 角色所建立的應用程式不受影響。不過，我們建議新增 IAM 角色以授予使用者存取其他帳戶中的資源，因為這樣可確保套用使用服務控制政策 (SCP) 的組織界限，以決定誰可以從組織中的帳戶傳送和接收事件。

> **Important**
>
> 如果您選擇接收來自所有 AWS 帳戶的事件，請小心建立只符合要從其他人接收的事件的規則。若要建立更多安全規則，請務必確認每個規則的事件模式包含 Account 欄位，此欄位為一或多個您想接收事件的帳戶，以及這些帳戶的 ID。具有包含 Account (帳戶) 欄位的事件模式的規則，不符合從 Account 欄位中未列出帳戶所傳送的事件。如需詳細資訊，請參閱 Amazon EventBridge 活動。

AWS 帳戶之間事件的規則

如果您的帳戶設定為接收來自其他 AWS 帳戶的事件匯流排的事件，您可以撰寫符合這些事件的規則。設定規則的事件模式，以符合您從其他帳戶所接收的事件。

除非您在規則的事件模式中指定 account，否則任何您帳戶中符合從其他帳戶內事件匯流排接收事件的新規則或現有規則，都將會根據這些事件而觸發。如果您正從其他帳戶接收事件，且您需要規則僅在規則從您自己的帳戶生成事件模式時觸發該事件模式，您必須新增 account 並指定自己的帳戶 ID 到規則的事件模式。

如果您將 AWS 帳戶設定為接受來自所有 AWS 帳戶中事件匯流排的事件，我們強烈建議您新增account至帳戶中的每個 EventBridge 規則。這樣可以防止您帳戶中的規則觸發來自未知 AWS 帳號的事件。當您在規則中指定 account 欄位時，您可以在該欄位中指定多個 AWS 帳戶的帳戶 ID。

若要針對您已授與權限的 AWS 帳戶中任何事件匯流排的相符事件觸發規則，請勿在規則account欄位中指定 *。這麼做就不會有任何事件符合，因為 * 絕不會出現在事件的 account 欄位中。反而會省略規則中的 account 欄位。

建立在 AWS 帳戶之間傳送事件的規則

將另一個帳戶中的事件匯流排指定為目標是建立規則的一部分。

使用主控台建立將事件傳送至不同 AWS 帳戶的規則

1. 然後依照 ??? 程序中的步驟進行操作。
2. 在 ??? 步驟中，當系統提示您選擇目標類型時：
   a. 選擇EventBridge 活動巴士。
   b. 選取不同帳戶或地區中的事件匯流排。
   c. 針對作為目標的事件匯流排，輸入您要使用的事件匯流排的 ARN。
3. 依照下列程序步驟，完成建立規則的操作。

在 AWS 區域之間傳送和接收 Amazon EventBridge 事件

您可以設 EventBridge 定在 AWS 區域之間傳送和接收事件。您也可以允許或拒絕來自特定區域的事件、與事件匯流排相關聯的特定規則，或來自特定來源的事件。如需詳細資訊，請參閱使用 Amazon 介紹跨區域事件路由 EventBridge

以下是支援的目的地區域：

- 非洲 (開普敦) 區域
- 亞太區域 (香港) 區域
- 亞太區域 (東京) 區域

- 亞太區域 (首爾) 區域
- 亞太 (大阪) 區域
- 亞太 (孟買) 區域
- 亞太區域 (新加坡) 區域
- 亞太區域 (雪梨) 區域
- 加拿大 (中部) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (斯德哥爾摩) 區域
- Europe (Milan) Region
- 歐洲 (愛爾蘭) 區域
- 歐洲 (倫敦) 區域
- 歐洲 (巴黎) 區域
- 中東 (阿拉伯聯合大公國) 區域
- Middle East (Bahrain) Region
- 南美洲 (聖保羅) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國東部 (俄亥俄) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 美國西部 (奧勒岡) 區域
- 亞太區域 (雅加達)
- 亞太區域 (墨爾本)
- 以色列 (特拉維夫) 區域

下列影片涵蓋使用 https://console.aws.amazon.com/events/、 AWS CloudFormation和 AWS Serverless Application Model：跨區域事件路由在區域之間路由事件的路由

建立將事件傳送至不同 AWS 區域的規則

將另一個 AWS 區域中的事件匯流排指定為目標是建立規則的一部分。

使用主控台建立將事件傳送至不同 AWS 帳戶的規則

1. 然後依照 ??? 程序中的步驟進行操作。
2. 在 ??? 步驟中，當系統提示您選擇目標類型時：
   a. 選擇EventBridge 活動巴士。
   b. 選取不同帳戶或地區中的事件匯流排。
   c. 針對作為目標的事件匯流排，輸入您要使用的事件匯流排的 ARN。
3. 依照下列程序步驟，完成建立規則的操作。

在同一帳戶和區域的事件匯流排之間傳送和接收 Amazon EventBridge 事件

您可以設 EventBridge 定在相同 AWS 帳戶和區域中的事件匯流排之間傳送和接收事件。

當您設定 EventBridge 為在事件匯流排之間傳送或接收事件時，您可以在傳送者事件匯流排上使用 IAM 角色，授與寄件者事件匯流排將事件傳送至接收者事件匯流排的權限。您可以在接收者事件匯流排上使用資源型政策，以授予接收者事件匯流排接收來自傳送者事件匯流排的許可。您也可以允許或拒絕來自特定事件匯流排的事件、與事件匯流排相關聯的特定規則，或來自特定來源的事件。如需有關事件匯流排許可的詳細資訊，包括範例政策，請參閱 Amazon EventBridge 事件匯流排的許可。

設定 EventBridge 在帳戶中事件匯流排之間傳送事件或接收事件的步驟如下：

- 若要使用現有的 IAM 角色，您需要將寄件者事件匯流排許可授予接收者事件匯流排，或將接收者事件匯流排許可授予寄件者事件匯流排。
- 針對寄件者事件匯流排，設定一個或多個將接收者帳戶的事件匯流排作為目標的規則，並建立 IAM 角色。如需應附加至角色之策略的範例，請參閱 ???。
- 在接收者事件匯流排上，編輯許可以允許從其他事件匯流排傳遞事件。
- 在接收者事件上，設定一個或多個符合來自寄件者事件匯流排之事件的規則。

> **Note**
>
> EventBridge 無法將從寄件者事件匯流排接收到的事件路由至第三個事件匯流排。

從一個事件匯流排傳送到另一個事件匯流排的事件會作為自訂事件。如需詳細資訊，請參閱 Amazon EventBridge 定價。

建立將事件傳送至相同 AWS 帳戶和區域中不同事件匯流排的規則

若要將事件傳送至另一個事件匯流排，您可以建立以事件匯流排做為目標的規則。在相同的 AWS 帳戶和區域中指定事件匯流排做為目標是建立規則的一部分。

若要使用主控台建立規則，將事件傳送至相同 AWS 帳戶中的不同事件匯流排，並使用 [區域]

1. 然後依照 ??? 程序中的步驟進行操作。
2. 在 ??? 步驟中，當系統提示您選擇目標類型時：
   a. 選擇EventBridge 活動巴士。
   b. 在同一 AWS 帳戶和區域中選擇事件總線。
   c. 針對事件匯流排作爲目標，從下拉式清單中選取事件匯流排。
3. 依照下列程序步驟，完成建立規則的操作。

Amazon EventBridge 輸入轉換

您可以在將資訊 EventBridge傳遞至規則目標之前，自訂事件中的文字。使用主控台或 API 中的輸入轉換器，您可以定義使用 JSON 路徑來參考原始事件來源中值的變數。轉換後的事件會傳送至目標，而不是原始事件。但是，動態路徑參數必須參照原始事件，而不是轉換的事件。您可以定義多達 100 個變數，從輸入中為每個變數指定一個值。然後，您可以在輸入範本中以 <*variable-name*> 形式使用這些變數。

如需使用輸入轉換器的教程，請參閱 ???。

> **Note**
>
> EventBridge 不支持所有 JSON 路徑語法，並在運行時對其進行評估。支援的語法包括：
>
> - 點符號 (例如 `$.detail`)
> - 破折號
> - 底線
> - 英數字元
> - 陣列索引
> - 萬用字元 (*)

在本主題中：

預定義變數

您可以在不定義 JSON 路徑的情況下使用預先定義的變數。這些變數會被保留，而且您無法使用這些名稱建立變數。

- `aws.events.rule-arn`— EventBridge 規則的 Amazon 資源名稱 (ARN)。
- `aws.events.rule-name`— 規 EventBridge 則的名稱。
- `aws.events.event.ingestion-time`— 事件被接收的時間 EventBridge。這是一個 ISO 8601 時間戳記。此變數由所產生， EventBridge 且無法覆寫。
- `aws.events.event`：原始事件承載為 JSON (不含 `detail` 欄位)。僅能用作 JSON 欄位的值，因為它的內容不會逸出。
- `aws.events.event.json`：完整的原始事件承載為 JSON (含 `detail` 欄位)。僅能用作 JSON 欄位的值，因為它的內容不會逸出。

輸入轉換範例

以下為範例 Amazon EC2 事件。

```
{
  "version": "0",
  "id": "7bf73129-1428-4cd3-a780-95db273d1602",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "123456789012",
  "time": "2015-11-11T21:29:54Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"
  ],
  "detail": {
    "instance-id": "i-0123456789",
    "state": "RUNNING"
  }
}
```

在主控台中定義規則時，請選取設定輸入下的輸入轉換器選項。此選項會顯示兩個文字方塊：一個用於 Input Path (輸入路徑)，另一個用於 Input Template (輸入範本)。

輸入路徑係用於定義變數。使用 JSON 路徑來參考事件中的項目，並將這些值存放在變數中。例如，您可以在第一個文字方塊中輸入下列內容，建立輸入路徑以參照範例事件中的值。您還可以使用括號和索引從陣列中獲取項目。

> **Note**
>
> EventBridge 在運行時替換輸入轉換器，以確保有效的 JSON 輸出。因此，請在參考 JSON 路徑參數的變數周圍加上引號，但不要在參考 JSON 物件或陣列的變數周圍加上引號。

```
{
  "timestamp" : "$.time",
  "instance" : "$.detail.instance-id",
  "state" : "$.detail.state",
  "resource" : "$.resources[0]"
}
```

這會定義四個變數，<timestamp>、<instance>、<state> 和 <resource>。您可以在建立輸入範本時參考這些變數。

輸入範本是您要傳遞到目標的資訊範本。您可以建立一個將字串或 JSON 傳遞給目標的範本。使用先前的事件和輸入路徑，下列輸入範本範例會在將事件路由至目標之前，將事件轉換為範例輸出。

描述	範本	輸出
簡單字串	`"instance <instance> is in <state>"`	`"instance i-0123456789 is in RUNNING"`
具有溢出引號的字串	`"instance \"<instance> \" is in <state>"`	`"instance \"i-0123456789\" is in RUNNING"` 請注意，這是控 EventBridge 制台中的行為。 AWS CLI 會溢出斜線字元，結果是 `"instance "i-0123456789" is in RUNNING"` 。
簡單 JSON	`{ "instance" : <instance>, "state": <state>`	`{ "instance" : "i-0123456789", "state": "RUNNING"`

描述	範本	輸出
	<pre>}</pre>	<pre>}</pre>

帶有字符串和變數的 JSON	<pre>{
	 "instance" : <instance
	>,
	 "state": "<state>",
	 "instanceStatus":
	 "instance \"<instance>
	\" is in <state>"
	}</pre>
		<pre>{
		 "instance" : "i-012345
		6789",
		 "state": "RUNNING",
		 "instanceStatus":
		 "instance \"i-01234
		56789\" is in RUNNING"
		}</pre>

JSON 與變數和靜態資訊的混合	<pre>{
	 "instance" :
	<instance>,
	 "state": [9, <state>,
	true],
	 "Transformed" : "Yes"
	}</pre>
		<pre>{
		 "instance" :
		"i-0123456789",
		 "state": [
		 9,
		 "RUNNING",
		 true
		],
		 "Transformed" : "Yes"
		}</pre>

在 JSON 中包含保留變數	<pre>{
	 "instance" :
	<instance>,
	 "state": <state>,
	 "ruleArn" : <aws.even
	ts.rule-arn>,
	 "ruleName" :
	<aws.events.rule-n
	ame>,
	 "originalEvent" :
	<aws.events.event.
	json>
	}</pre>
		<pre>{
		 "instance" :
		"i-0123456789",
		 "state": "RUNNING",
		 "ruleArn" : "arn:aws:
		events:us-east-2:1
		23456789012:rule/e
		xample",
		 "ruleName" :
		"example",
		 "originalEvent" : {
		 ... // commented for
		brevity
		 }
		}</pre>

描述	範本	輸出
在字串中包含保留變數	`"<aws.events.rule-name> triggered"`	`"example triggered"`
Amazon CloudWatch 日誌組	`{ "timestamp" : <timestamp>, "message": "instance \"<instance>\" is in <state>" }`	`{ "timestamp" : 2015-11-11T21:29:54Z, "message": "instance "i-0123456789" is in RUNNING }`

使用 EventBridge API 轉換輸入

如需有關使用 EventBridge API 轉換輸入的資訊，請參閱使用輸入轉換器從事件擷取資料，並將該資料輸入至目標。

通過使用轉換輸入 AWS CloudFormation

如需有關使用轉 AWS CloudFormation 換輸入的資訊，請參閱AWS::Events::Rule InputTransformer。

轉換輸入的常見問題

以下是轉換輸入時的一些常見問題 EventBridge：

- 針對字串，引號是必要的。
- 為您的範本建立 JSON 路徑時沒有驗證。
- 如果您指定變數對應不存在於事件中的 JSON 路徑，則該變數不會建立，且不會出現在輸出中。
- 像 `aws.events.event.json` 的 JSON 屬性只能用作 JSON 欄位的值，而不能內嵌在其他字串中。
- EventBridge 填入目標的輸入範本時，不會逸出由輸入路徑擷取的值。
- 如果 JSON 路徑參考 JSON 物件或陣列，但該變數在字串中參照，則 EventBridge 會移除任何內部引號以確保有效的字串。例如，對於`<detail>`指向的變數`$.detail`，「Detail is<detail>」會導致從物件中 EventBridge 移除引號。

因此，如果您想要根據單一 JSON 路徑變數來輸出 JSON 物件，則必須將其做為索引鍵放置。在此範例中，`{"detail": <detail>}`。

- 代表字串的變數不需要引號。它們是允許的，但在轉換過程中 EventBridge 會自動將引號添加到字符串變量值中，以確保轉換輸出是有效的 JSON。 EventBridge 不會在代表 JSON 物件或陣列的變數中加入引號。請勿為代表 JSON 物件或陣列的變數加上引號。

  例如，下列輸入範本包含代表字串和 JSON 物件的變數：

  ```
  {
    "ruleArn" : <aws.events.rule-arn>,
    "ruleName" : <aws.events.rule-name>,
    "originalEvent" : <aws.events.event.json>
  }
  ```

  使用適當的引號導致有效的 JSON：

  ```
  {
    "ruleArn" : "arn:aws:events:us-east-2:123456789012:rule/example",
    "ruleName" : "example",
    "originalEvent" : {
      ... // commented for brevity
    }
  }
  ```

- 對於（非 JSON）文本輸出為多行字符串，請用雙引號將輸入模板中的每個單獨的行包裹起來。

  例如，如果您將「Amazon Inspector 尋找」事件與下列事件模式進行比對：

  ```
  {
    "detail": {
      "severity": ["HIGH"],
      "status": ["ACTIVE"]
    },
    "detail-type": ["Inspector2 Finding"],
    "source": ["inspector2"]
  }
  ```

  並使用以下輸入路徑：

  ```
  {
  ```

```
  "account": "$.detail.awsAccountId",
  "ami": "$.detail.resources[0].details.awsEc2Instance.imageId",
  "arn": "$.detail.findingArn",
  "description": "$.detail.description",
  "instance": "$.detail.resources[0].id",
  "platform": "$.detail.resources[0].details.awsEc2Instance.platform",
  "region": "$.detail.resources[0].region",
  "severity": "$.detail.severity",
  "time": "$.time",
  "title": "$.detail.title",
  "type": "$.detail.type"
}
```

您可以使用下面的輸入模板來生成多行字符串輸出：

```
"<severity> severity finding <title>"
"Description: <description>"
"ARN: \"<arn>\""
"Type: <type>"
"AWS Account: <account>"
"Region: <region>"
"EC2 Instance: <instance>"
"Platform: <platform>"
"AMI: <ami>"
```

將輸入轉換器設定為建立規則的一部分

在建立規則時，您可以指定輸入轉換器，以 EventBridge 便在將這些事件傳送至指定目標之前處理相符事件。您可以為 AWS 服務或 API 目的地的目標設定輸入轉換器。

建立目標輸入轉換器做為規則的一部分

1. 請遵循建立規則的步驟，如 ??? 中所述。
2. 在步驟 3 - 選取目標中，展開其他設定。
3. 針對設定目標輸入，從下拉式清單中選擇輸入轉換器。

   按一下設定輸入轉換器。

   EventBridge 顯示「規劃輸入轉換器」對話方塊。

4. 在範例事件區段中，選擇您要測試事件模式的範例事件類型。您可以選擇 AWS 活動、合作夥伴活動，或輸入您自己的自訂活動。

   AWS events

   從支援 AWS 服務發出的事件中選取。

   1. 選取 AWS 事件。
   2. 在「範例事件」下，選擇所需的 AWS 事件。活動按 AWS 服務組織。

      當您選取事件時， EventBridge 會填入範例事件。

      例如，如果您選擇 S3 物件已建立，則 EventBridge 會顯示 S3 物件建立事件的範例。
   3. (選用) 您也可以選取複製，將範例事件複製到裝置的剪貼簿。

   Partner events

   從支援 EventBridge的協力廠商服務 (例如 Salesforce) 發出的事件中選取。

   1. 選取EventBridge 合作夥伴活動。
   2. 在範例事件下，選擇所需的合作夥伴事件。事件由合作夥伴進行組織。

      當您選取事件時， EventBridge 會填入範例事件。
   3. (選用) 您也可以選取複製，將範例事件複製到裝置的剪貼簿。

   Enter your own

   以 JSON 文字輸入您自己的事件。

   1. 選取輸入您自己的。
   2. EventBridge 使用必要事件屬性的範本填入範例事件。
   3. 視需要編輯並新增至範例事件。範例事件必須是有效的 JSON。
   4. (選用) 您也可以選擇下列任一選項：
      - 複製：將範例事件複製到裝置的剪貼簿。
      - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。
5. (選用) 展開範例輸入路徑、範本和輸出區段，以查看下列範例：

- 如何使用 JSON 路徑定義代表事件資料的變數
- 如何在輸入轉換器模板中使用這些變量
- EventBridge 發送到目標的結果輸出

如需輸入轉換的詳細範例，請參閱 ???。

6. 在目標輸入轉換器區段中，定義要在輸入範本中使用的任何變數。

   使用 JSON 路徑來參考原始事件來源中值的變數。然後，您可以在輸入範本中參考這些變數，以便在 EventBridge 傳遞至目標的轉換事件中包含來自原始來源事件的資料。您最多可定義 100 個變數。輸入轉換器必須是有效的 JSON。

   例如，假設您已選擇 AWS 事件 S3 物件已建立作為此輸入轉換器的範例事件。然後，您可以定義在模板中使用的以下變量：

   ```
   {
     "requester": "$.detail.requester",
     "key": "$.detail.object.key",
     "bucket": "$.detail.bucket.name"
   }
   ```

   (選用) 您也可以選擇複製，將輸入轉換器複製到裝置的剪貼簿。

7. 在「範本」(Template) 區段中，構成您要使用的範本來決定 EventBridge 傳遞至目標的範本。

   您可以使用 JSON、字符串、靜態信息、您定義的變量以及保留變量。如需輸入轉換的詳細範例，請參閱 ???。

   例如，假設您已經在上一個範例中定義了變數。然後，您可以編寫以下模板，該模板引用這些變量以及保留變量和靜態信息。

   ```
   {
       "message": "<requester> has created the object \"<key>\" in the bucket
    \"<bucket>\"",
       "RuleName": <aws.events.rule-name>,
       "ruleArn" : <aws.events.rule-arn>,
       "Transformed": "Yes"
   }
   ```

   (選用) 您也可以選擇複製，將範本複製到裝置的剪貼簿。

8. 若要測試範本，請選取產生輸出。

   EventBridge 根據輸入範本處理範例事件，並顯示在「輸出」(Output) 下產生的轉換後的輸出。這是 EventBridge 將傳遞給目標以取代原始來源事件的資訊。

   上述範例輸入範本所產生的輸出如下：

   ```
   {
       "message": "123456789012 has created the object "example-key" in the bucket
    "example-bucket"",
       "RuleName": rule-name,
       "ruleArn" : arn:aws:events:us-east-1:123456789012:rule/rule-name,
       "Transformed": "Yes"
   }
   ```

   (選用) 您也可以選擇複製，將產生的輸出複製到裝置的剪貼簿。

9. 選取確認。
10. 請遵循建立規則的其餘步驟，如 ??? 中所述。

使用 EventBridge 沙箱測試目標輸入變壓器

在將資訊 EventBridge傳遞至規則目標之前，您可以使用輸入轉換器自訂事件中的文字。

設定輸入轉換器通常是在建立新規則或編輯現有規則時指定目標的較大程序的一部分。不過 EventBridge，使用中的沙箱，您可以快速設定輸入轉換器，並使用範例事件來確認您取得所需的輸出，而不必建立或編輯規則。

如需有關輸入轉型的詳細資訊，請參閱 ???。

測試目標輸入轉換器

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在開發人員資源下，選擇沙盒，然後在沙盒頁面上選擇目標輸入轉換器標籤。
3. 在範例事件區段中，選擇您要測試事件模式的範例事件類型。您可以選擇 AWS 活動、合作夥伴活動，或輸入您自己的自訂活動。

   AWS events

   從支援 AWS 服務發出的事件中選取。

1. 選取 AWS 事件。
2. 在「範例事件」下，選擇所需的 AWS 事件。活動按 AWS 服務組織。

   當您選取事件時， EventBridge 會填入範例事件。

   例如，如果您選擇 S3 物件已建立，則 EventBridge 會顯示 S3 物件建立事件的範例。
3. (選用) 您也可以選取複製，將範例事件複製到裝置的剪貼簿。

Partner events

從支援 EventBridge的協力廠商服務 (例如 Salesforce) 發出的事件中選取。

1. 選取EventBridge 合作夥伴活動。
2. 在範例事件下，選擇所需的合作夥伴事件。事件由合作夥伴進行組織。

   當您選取事件時， EventBridge 會填入範例事件。
3. (選用) 您也可以選取複製，將範例事件複製到裝置的剪貼簿。

Enter your own

以 JSON 文字輸入您自己的事件。

1. 選取輸入您自己的。
2. EventBridge 使用必要事件屬性的範本填入範例事件。
3. 視需要編輯並新增至範例事件。範例事件必須是有效的 JSON。
4. (選用) 您也可以選擇下列任一選項：
   - 複製：將範例事件複製到裝置的剪貼簿。
   - 美化：透過新增換行符號、定位鍵和空格鍵，讓 JSON 文字更易於閱讀。

4. (選用) 展開範例輸入路徑、範本和輸出區段，以查看下列範例：

- 如何使用 JSON 路徑定義代表事件資料的變數
- 如何在輸入轉換器模板中使用這些變量
- EventBridge 發送到目標的結果輸出

如需輸入轉換的詳細範例，請參閱 ???。

5. 在目標輸入轉換器區段中，定義要在輸入範本中使用的任何變數。

   使用 JSON 路徑來參考原始事件來源中值的變數。然後，您可以在輸入範本中參考這些變數，以便在 EventBridge 傳遞至目標的轉換事件中包含來自原始來源事件的資料。您最多可定義 100 個變數。輸入轉換器必須是有效的 JSON。

   例如，假設您已選擇 AWS 事件 S3 物件已建立作為此輸入轉換器的範例事件。然後，您可以定義在模板中使用的以下變量：

   ```
   {
     "requester": "$.detail.requester",
     "key": "$.detail.object.key",
     "bucket": "$.detail.bucket.name"
   }
   ```

   (選用) 您也可以選擇複製，將輸入轉換器複製到裝置的剪貼簿。

6. 在「範本」(Template) 區段中，構成您要使用的範本來決定 EventBridge 傳遞至目標的範本。

   您可以使用 JSON、字符串、靜態信息、您定義的變量以及保留變量。如需輸入轉換的詳細範例，請參閱 ???。

   例如，假設您已經在上一個範例中定義了變數。然後，您可以編寫以下模板，該模板引用這些變量以及保留變量和靜態信息。

   ```
   {
       "message": "<requester> has created the object \"<key>\" in the bucket
    \"<bucket>\"",
       "RuleName": <aws.events.rule-name>,
       "ruleArn" : <aws.events.rule-arn>,
       "Transformed": "Yes"
   }
   ```

   (選用) 您也可以選擇複製，將範本複製到裝置的剪貼簿。

7. 若要測試範本，請選取產生輸出。

   EventBridge 根據輸入範本處理範例事件，並顯示在「輸出」(Output) 下產生的轉換後的輸出。這是 EventBridge 將傳遞給目標以取代原始來源事件的資訊。

   上述範例輸入範本所產生的輸出如下：

```
{
    "message": "123456789012 has created the object "example-key" in the bucket
 "example-bucket"",
    "RuleName": rule-name,
    "ruleArn" : arn:aws:events:us-east-1:123456789012:rule/rule-name,
    "Transformed": "Yes"
}
```

(選用) 您也可以選擇複製，將產生的輸出複製到裝置的剪貼簿。

Amazon EventBridge 封存和重播

在 EventBridge 中，您可以建立事件的封存，以便日後輕鬆地重播這些事件。例如，您可能想要重播事件以從錯誤中還原或驗證應用程式中的新功能。

> **Note**
>
> 將事件發佈至活動匯流排與到達到封存的活動之間可能會有延遲。我們建議您延遲重播封存的活動 10 分鐘，以確保所有事件都能重播。

以下視頻演示了存檔和重播的用法：建立封存和重播

主題

- 封存 Amazon EventBridge 事件
- 重播已封存的 Amazon EventBridge 事件

封存 Amazon EventBridge 事件

當您在 EventBridge 中建立封存時，您可以透過指定事件模式來判斷哪些事件要傳送至封存。EventBridge 會將符合事件模式的事件傳送至封存。您也可以設定保留期間，以在將其捨棄之前將事件儲存於封存中。

預設情況下，EventBridge 在 AWS 擁有的 CMK 下使用 256 位元進階加密標準 (AES-256) 來加密封存中的事件資料，這有助於保護您的資料免遭未經授權的存取。

> **Note**
>
> 過期的事件通常會每 24 小時從 `DescribeArchive` 作業的 `EventCount` 和 `SizeBytes` 值中進行扣除。因此，最近過期的事件可能不會反映在這些值中。

建立所有事件的封存

1. 造訪 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在左側導覽窗格中，選擇封存。
3. 選擇建立封存。
4. 在封存詳細資訊下，輸入封存的名稱。在所選的區域中，此名稱對於您的賬戶必須是唯一的。

   建立封存後無法修改名稱。
5. (選用) 輸入針對封存的說明。
6. 對於來源，選取發出要傳送至封存之事件的事件匯流排。
7. 對於保留期，執行下列任意一項：
   - 選擇無限期可將事件保留於封存中，而不會刪除它們。
   - 輸入要保留事件的天數。在指定天數之後，EventBridge 會從封存中刪除這些事件。
8. 選擇 下一步。
9. 在事件模式下，選擇無事件篩選。
10. 選擇建立封存。

建立具有事件模式的封存

1. 造訪 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在左側導覽窗格中，選擇封存。

3. 選擇建立封存。
4. 在封存詳細資訊下，輸入封存的名稱。在所選的區域中，此名稱對於您的賬戶必須是唯一的。

   建立封存後無法修改名稱。
5. (選用) 輸入針對封存的說明。
6. 對於來源，選取發出要傳送至封存之事件的事件匯流排。
7. 對於保留期，執行下列任意一項：
   - 選擇無限期* 可將事件保留於封存中，而不會刪除它們。
   - 輸入要保留事件的天數。在指定天數之後，EventBridge 會從封存中刪除這些事件。
8. 選擇 下一步。
9. 在事件模式下，選擇按事件模式比對篩選事件。
10. 執行下列任意一項：
    - 選取模式建置器，然後選取服務供應商。如果您選取 AWS，也請選取要在模式中使用的 AWS 服務名稱和事件類型。
    - 選取 JSON 編輯器以手動建立模式。您也可以從某項規則中複製模式，然後將其貼上至 JSON 編輯器內。
11. 選取建立封存。

若要確認事件已成功傳送至封存，您可以使用 EventBridge API 的 `DescribeArchive` 作業來查看 `EventCount` 是否反映封存中的事件數目。如果事件數目為 0，則封存中無任何事件。

重播已封存的 Amazon EventBridge 事件

建立封存之後，您就可以重播封存中的事件。例如，如果您使用其他功能更新應用程式，您可以重播歷史事件，以確保事件會重新處理，以保持應用程式的一致性。您也可以使用封存重播新功能的事件。當您重播事件時，您可以指定要從哪個封存重播事件、要重播的事件開始和結束時間、事件匯流排或一個或多個要重播事件所遵循的規則。

事件不必以新增至封存的相同順序進行重播。重播會根據事件中的時間來處理要重播的事件，並以一分鐘的間隔重播事件。如果您指定的事件開始時間和結束時間涵蓋 20 分鐘的時間範圍，則會先從該 20 分鐘範圍的第一分鐘開始重播事件。然後，重播第二分鐘的事件。您可以使用 EventBridge API 的 `DescribeReplay` 作業來判斷重播的進度。`EventLastReplayedTime` 傳回重播之上個事件的時間戳記。

事件會基於 (但不同於) AWS 帳戶的每秒 `PutEvents` 交易數限制進行重播。您可以請求提高針對 PutEvents 的限制。如需詳細資訊，請參閱 Amazon EventBridge 配額。

> **Note**
>
> 每個 AWS 區域每個帳戶最多可以有 10 個作用中的并發重播。

若要開始事件重播

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在左側導覽窗格中，請選擇重播。
3. 選擇開始新的重播。
4. 輸入重播的名稱，或者輸入描述。
5. 對於來源，選取要從中重播事件的封存。
6. 對於目的地，您可以將事件僅重播至發出事件的相同事件匯流排。
7. 對於指定規則，請執行下列其中一個動作：
   - 選擇所有規則，將事件重播至所有規則
   - 選擇指定規則，然後選取要重播事件的一個或多個規則。
8. 在重播時間範圍下，指定日期、時間以及開始時間和結束時間的時區。僅會重播在開始時間和結束時間之間發生的事件。
9. 選擇開始重播。

重播已封存的事件時，重播的狀態為已完成。

如果您開始重播並想要中斷它，僅要狀態為開始或執行中，您就可以取消它。

取消重播

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在左側導覽窗格中，請選擇重播。
3. 選擇要取消的重播。
4. 選擇取消。

Amazon EventBridge Pipes

Amazon EventBridge 管道將來源連接到目標。管道用於支援的來源和目標之間的點對點整合，並支援進階轉換和擴充。在開發事件驅動的架構時，它可減少對專業知識和整合程式碼的需求，並促進公司應用程式的一致性。若要設定管道，您可以選擇來源、新增可選篩選、定義可選的擴充，以及選擇事件資料的目標。

> **Note**
>
> 您也可以使用事件匯流排來路由活動。事件匯流排非常適合事件驅動服務之間的多對多路由事件。如需更多詳細資訊，請參閱 ???。

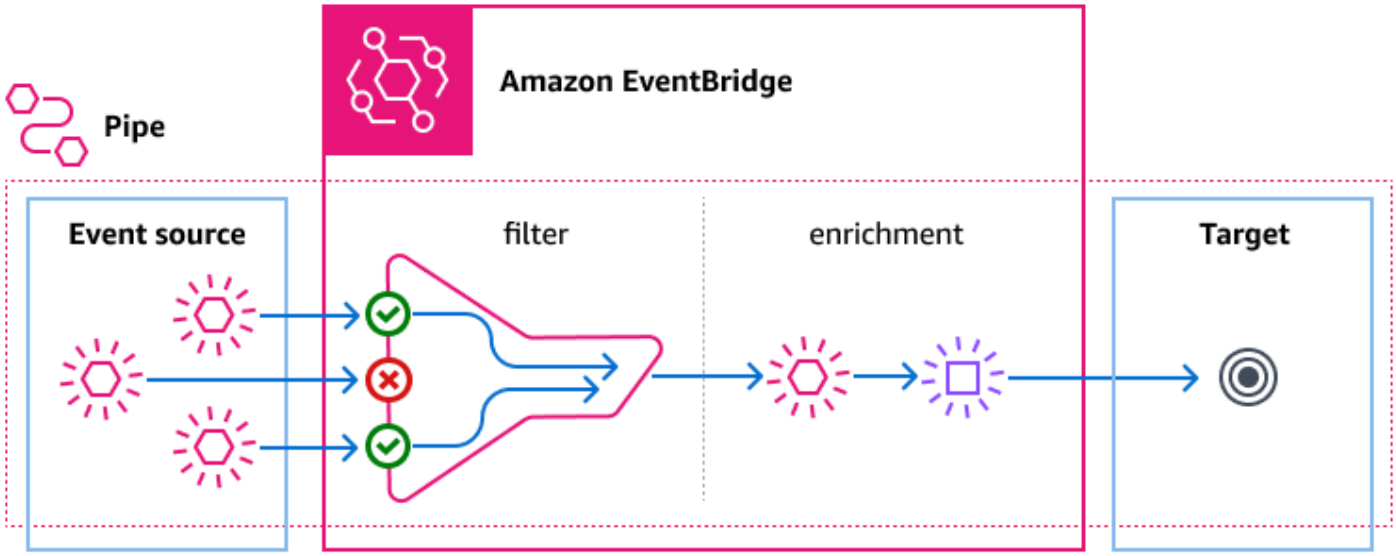
EventBridge 管道如何工作

在高階程序中，EventBridge 管道的運作方式如下：

1. 您在帳戶中建立管道。其中包含：
   - 指定您希望管道接收事件的其中一個受支援的事件來源。
   - 或者，規劃篩選器，以便管道僅處理從來源接收到的事件子集。
   - 選擇性地設定擴充步驟，以在將事件資料傳送至目標之前增強事件資料。
   - 指定您希望管道向其中一個支援的目標傳送事件。
2. 事件來源會開始將事件傳送至管道，而管道會先處理事件，再將事件傳送至目標。
   - 如果您已設定篩選器，管道會評估事件，並且僅在與該篩選器相符時才將其傳送至目標。

     您只需為符合篩選條件的事件付費。
   - 如果您已設定擴充，則管道會在將事件傳送至目標之前對事件執行該擴充。

     如果事件是批次處理，則擴充會維護批次中事件的順序。



例如，管道可用於創建電子商務系統。假設您有包含客戶資訊的 API，例如運送地址。

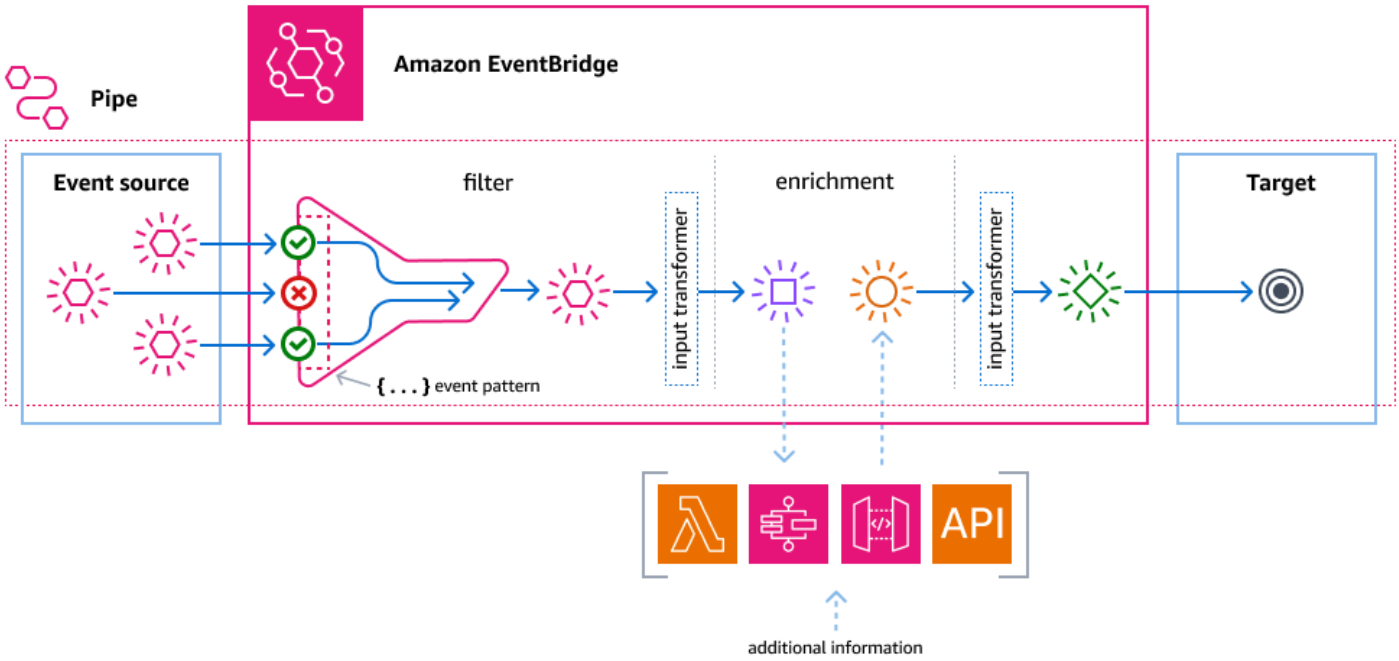
1. 那麼您可以執行下列操作來建立管道：
   - Amazon SQS 訂單收到訊息佇列作為事件來源。
   - 將 EventBridge 接 API 目標作為擴充項目
   - 作為目標的 AWS Step Functions 狀態機
2. 接著，當 Amazon SQS 訂單收到的訊息出現在佇列中時，訊息就會傳送至您的管道。
3. 接著，管道會將該資料傳送至 EventBridge API 目的地擴充功能，此功能會傳回該訂單的客戶資訊。
4. 最後，管道將擴充的數據發送到 AWS Step Functions 狀態機，該狀態機處理訂單。

EventBridge 管道概念

以下是 EventBridge 管道的基本元件進一步了解。

管道

管道會將事件從單一來源路由至單一目標。管道還包括篩選特定事件的功能，以及在事件資料傳送至目標之前對事件資料執行擴充。



來源

EventBridge 管道會接收來自各種來源的事件資料、將選用的篩選器和擴充套用至該資料，然後將其傳送至目標。如果來源對傳送至管道的事件強制執行順序，則該順序會在整個程序中保留至目標。

如需來源的詳細資訊，請參閱 ???。

篩選條件

一個管道可以篩選指定來源的事件，並僅處理其中的一個子集。若要在管道上配置篩選，您可以定義管道使用的事件模式來決定要傳送至目標的事件。

您只需為符合篩選條件的事件付費。

如需更多詳細資訊，請參閱 ???。

擴充

透過 EventBridge 管道的擴充步驟，您可以在將來源資料傳送到目標之前先增強來源的資料。例如，您可能會收到不包含完整工單資料的票證建立的事件。使用擴充，您可以有一個 Lambda 函數呼叫 `get-ticket` API 以獲取完整的工單詳細信息。然後管道可以將該資訊傳送至目標。

如需事件資訊的詳細資料，請參閱 ???。

目標

篩選和擴充事件資料之後，您可以指定將其傳送到特定目標的管道，例如 Amazon Kinesis 串流或 Amazon CloudWatch 日誌群組。如需可用目標的清單，請參閱 ???。

您可以在資料增強之後以及管道傳送至目標之前對其進行轉換。如需更多詳細資訊，請參閱 ???。

多個 Pipes (每個 Pipe 都有不同的來源) 都可以將事件傳送到相同的目標。

您也可以同時使用管道和事件匯流排，將事件傳送至多個目標。常見的使用案例是建立以事件匯流排作為其目標的管道；管道會將事件傳送至事件匯流排，然後將這些事件傳送至多個目標。例如，您可以建立一個管道，其中包含來源的 DynamoDB 串流，並建立事件匯流排做為目標。管道會從 DynamoDB 串流接收事件，並將它們傳送至事件匯流排，然後根據您在事件匯流排上指定的規則，將事件傳送至多個目標。

Amazon EventBridge Pipes 的許可

設定管道時，您可以使用現有的執行角色，或讓 EventBridge 為您建立具有所需許可的執行角色。EventBridge 管道所需的許可會根據來源類型而有所不同，如下所示。如果您要設定自己的執行角色，則必須自行新增這些許可。

> **Note**
>
> 如果您不確定存取來源所需的確切範圍適當的許可，請使用 EventBridge Pipes 主控台建立新角色，然後檢查原則中列出的動作。

主題

- DynamoDB 執行角色許可
- Kinesis 執行角色許可
- Amazon MQ 執行角色許可
- Amazon MSK 執行角色許可
- 自我管理的 Apache Kafka 執行角色許可
- Amazon SQS 執行角色許可
- 擴充和目標許可

DynamoDB 執行角色許可

若是 DynamoDB 串流，EventBridge 管道需要下列許可，才能管理與 DynamoDB 資料串流相關的資源。

- dynamodb:DescribeStream
- dynamodb:GetRecords
- dynamodb:GetShardIterator
- dynamodb:ListStreams

若要將失敗批次的記錄傳送至管道無效字母佇列，您的管道執行角色需要下列許可：

- sqs:SendMessage

Kinesis 執行角色許可

若是 Kinesis，EventBridge 管道需要下列許可，才能管理與 Kinesis 資料串流相關的資源。

- kinesis:DescribeStream
- kinesis:DescribeStreamSummary
- kinesis:GetRecords
- kinesis:GetShardIterator
- kinesis:ListShards
- kinesis:ListStreams
- kinesis:SubscribeToShard

若要將失敗批次的記錄傳送至管道無效字母佇列，您的管道執行角色需要下列許可：

- sqs:SendMessage

Amazon MQ 執行角色許可

若是 Amazon MQ，EventBridge 管道需要下列許可，才能管理與 Amazon MQ 訊息代理程式相關的資源。

- mq:DescribeBroker
- secretsmanager:GetSecretValue
- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcs
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

Amazon MSK 執行角色許可

若是 Amazon MSK，EventBridge 管道需要下列許可，才能管理與 Amazon MSK 主題相關的資源。

> **Note**
>
> 如果您使用 IAM 角色型身份驗證，您的執行角色除了下面列出的許可之外，還需要 ??? 列出的許可。

- kafka:DescribeClusterV2
- kafka:GetBootstrapBrokers
- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

自我管理的 Apache Kafka 執行角色許可

對於自我管理的 Apache Kafka，EventBridge 需要下列權限來管理與您自我管理的 Apache Kafka 資料流相關的資源。

所需的許可

若要在 Amazon CloudWatch Logs 中建立日誌並存放在日誌群組中，您的 管道在其執行角色中必須具有下列許可：

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

可選的許可。

您的管道可能需要許可，才能：

- 描述您 Secrets Manager 機密。
- 存取您的 AWS Key Management Service (AWS KMS) 客戶受管金鑰。
- 存取 Amazon VPC。

Secrets Manager 和 AWS KMS 許可

視您為 Kafka 代理程式設定的存取控制類型而定，您的管道可能需要許可來存取您的 Secrets Manager 機密或解密您的 AWS KMS 客戶受管金鑰。若要連線至這些資源，函數的執行角色必須具有下列許可：

- secretsmanager:GetSecretValue
- kms:Decrypt

VPC 許可

如果只有某個 VPC 內的使用者可以存取自我管理 Apache Kafka 叢集，則您的管道必須具有存取 Amazon VPC 資源的許可。這些資源包括您的 VPC、子網路、安全群組和網路界面。若要連線至這些資源，管道的執行角色必須具有下列許可：

- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups

Amazon SQS 執行角色許可

若是 Amazon SQS，EventBridge 需要下列許可，才能管理與 Amazon SQS 佇列相關的資源。

- sqs:ReceiveMessage
- sqs:DeleteMessage
- sqs:GetQueueAttributes

擴充和目標許可

為了能夠根據您所擁有的資源進行 API 呼叫，EventBridge 管道需要適當的許可。EventBridge 管道會使用您在管道上指定的 IAM 角色，以便使用 IAM 主體 `pipes.amazonaws.com` 進行擴充和鎖定目標呼叫。

創建一個 Amazon EventBridge 管道

EventBridge 管道可讓您建立來源和目標之間的 point-to-point 整合，包括進階事件轉換和擴充。若要建立 EventBridge 管路，請執行下列步驟：

1. ???
2. ???
3. ???
4. ???
5. ???

如需如何使用 CLI 建立管道的相關資訊，請參閱 AWS CLI 命令參考中的建立管道。AWS

指定來源

若要開始，請指定管道接收事件的來源。

使用控制台指定管道來源的步驟

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇管道。
3. 選擇建立管道。
4. 輸入管道的名稱。
5. (選用) 輸入管道的描述。
6. 在建立管道頁籤上，針對來源，選擇要為此管指定的來源類型，然後規劃來源。

   組態屬性會因您選擇的來源類型而有所不同：

   Confluent

   若要將 Confluent 雲端串流設定為來源，請使用主控台

   1. 對於「來源」，請選擇「統一雲端」。
   2. 針對引導伺服器，輸入您的經紀人的 `host:port` 對地址。
   3. 在主題名稱中，輸入管道將從中讀取的主題名稱。
   4. (選用) 針對 VPC，選擇您要的 VPC。然後，針對 VPC 子網路，選擇所需的子網路。針對 VPC 安全群組，請選擇安全群組。
   5. 對於驗證-選用，請開啟使用驗證，然後執行下列操作：
      a. 針對驗證方法，請選擇驗證類型。
      b. 針對秘密金鑰，請選擇秘密金鑰。

      如需詳細資訊，請參閱 Confluent 文件中的驗證以統一雲端資源。
   6. (選用) 針對其他設定，請執行下列動作：
      a. 針對起始位置，選擇下列的一或多個選項：
         - 最新：使用碎片中的最新記錄開始讀取串流。
         - 修剪水平線：使用碎片中最後一個未修剪的記錄開始讀取串流。這是碎片中最古老的記錄。
      b. 針對批次大小 - 選用，輸入每個批次的最大記錄數。預設值為 100.
      c. 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。

DynamoDB

1. 針對來源，選擇 DynamoDB。
2. 針對 DynamoDB 串流，請選擇您要用作來源的串流。
3. 針對起始位置，選擇下列的一或多個選項：
   - 最新：使用碎片中的最新記錄開始讀取串流。
   - 修剪水平線：使用碎片中最後一個未修剪的記錄開始讀取串流。這是碎片中最古老的記錄。
4. (選用) 針對其他設定，請執行下列動作：
   a. 針對批次大小 - 選用，輸入每個批次的最大記錄數。預設值為 100.
   b. 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。
   c. 針對每個碎片的並行批次 - 選用 ，輸入可同時讀取的相同碎片中的批次數。
   d. 針對部份批次料號失敗時，請選擇下列項目：
      - AUTOMATIC_BISECT：將每個批次減半並分別重試，直到處理完所有記錄或批次中剩下一則失敗訊息為止。

> **Note**
>
> 如果您沒有選擇 AUTOMATIC_BISECT，您可以傳回特定失敗的記錄，而且只有那些記錄會重試。

Kinesis

若要使用主控台設定 Kinesis 來源

1. 針對來源，選擇 Kinesis。
2. 針對 Kinesis 串流，請選擇您要用作來源的串流。
3. 針對起始位置，選擇下列的一或多個選項：
   - 最新：使用碎片中的最新記錄開始讀取串流。
   - 修剪水平線：使用碎片中最後一個未修剪的記錄開始讀取串流。這是碎片中最古老的記錄。

- 在時間戳記：從指定的時間開始讀取串流。在時間戳記下，使用 YYYY/MM/DD 和 HH:mm: ss 格式輸入資料和時間。

4. (選用) 針對 Additional settings (其他設定)，執行下列動作：
   a. 針對批次大小 - 選用，輸入每個批次的最大記錄數。預設值為 100.
   b. (選用) 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。
   c. 針對每個碎片的並行批次 - 選用 ，輸入可同時讀取的相同碎片中的批次數。
   d. 針對部份批次料號失敗時，請選擇下列項目：
      - AUTOMATIC_BISECT：將每個批次減半並分別重試，直到處理完所有記錄或批次中剩下一則失敗訊息為止。

> **Note**
>
> 如果您沒有選擇 AUTOMATIC_BISECT，您可以傳回特定失敗的記錄，而且只有那些記錄會重試。

Amazon MQ

若要使用主控台設定 Amazon MQ 來源

1. 針對來源，選擇 Amazon MQ。
2. 針對 Amazon MQ 中介，請選擇您要用作來源的串流。
3. 在佇列名稱中，輸入管道將從中讀取的佇列名稱。
4. 針對驗證方法，請選擇基本驗證。
5. 針對秘密金鑰，請選擇秘密金鑰。
6. (選用) 針對其他設定，請執行下列動作：
   a. 針對批次大小 - 選用，輸入每個批次的最大訊息數。預設值為 100.
   b. 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。

Amazon MSK

若要使用主控台設定 Amazon MSK 來源

1. 針對來源，選擇 Amazon MSK。
2. 針對 Amazon MSK 叢集，選擇要開啟的叢集。

3. 在主題名稱中，輸入管道將從中讀取的主題名稱。
4. (選用) 針對取用者群組 ID，輸入要加入的取用者群組 ID。
5. (選用) 針對驗證 - 選用，請開啟使用驗證並執行下列動作：
   a. 針對驗證方法，選擇您想要的類型。
   b. 針對秘密金鑰，請選擇秘密金鑰。
6. (選用) 針對其他設定，請執行下列動作：
   a. 針對批次大小 - 選用，輸入每個批次的最大記錄數。預設值為 100.
   b. 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。
   c. 針對起始位置，選擇下列的一或多個選項：
      - 最新：使用碎片中最新記錄開始閱讀主題。
      - 修剪水平線：使用碎片中最後一個未修剪的記錄開始讀取主題。這是碎片中最古老的記錄。

> **Note**
>
> 針對 Apache Kafka，修剪地平線與最早是相同的。

Self managed Apache Kafka

若要使用主控台設定自我管理的 Apache Kafka 來源

1. 針對源，選擇自我管理 Apache Kafka。
2. 針對引導伺服器，輸入您的經紀人的 `host:port` 對地址。
3. 在主題名稱中，輸入管道將從中讀取的主題名稱。
4. (選用) 針對 VPC，選擇您要的 VPC。然後，針對 VPC 子網路，選擇所需的子網路。針對 VPC 安全群組，請選擇安全群組。
5. (選用) 針對驗證 - 選用，請開啟使用驗證並執行下列動作：
   a. 針對驗證方法，請選擇驗證類型。
   b. 針對秘密金鑰，請選擇秘密金鑰。
6. (選用) 針對其他設定，請執行下列動作：
   a. 針對起始位置，選擇下列的一或多個選項：
      - 最新：使用碎片中的最新記錄開始讀取串流。

- 修剪水平線：使用碎片中最後一個未修剪的記錄開始讀取串流。這是碎片中最古老的記錄。

b. 針對批次大小 - 選用，輸入每個批次的最大記錄數。預設值為 100.

c. 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。

Amazon SQS

若要使用主控台設定 Amazon SQS 來源

1. 針對來源，選擇 SQS。
2. 針對 SQS 佇列，請選擇要使用的佇列。
3. (選用) 針對其他設定，請執行下列動作：
   a. 針對批次大小 - 選用，輸入每個批次的最大記錄數。預設值為 100.
   b. 若為批次視窗 - 選用，請輸入在繼續之前收集記錄的秒數上限。

設定事件篩選 (選用)

您可以將篩選新增至管道，以便只將一部分事件從來源傳送至目標。

若要使用主控台設定篩選條件

1. 選擇篩選。
2. 在範例事件 - 選用下，您會看到可用來建立事件模式的範例事件，或者您可以選擇輸入您自己的事件來輸入您自己的事件。
3. 在事件模式底下，輸入您要用來篩選事件的事件模式。如需建構篩選器的更多資訊，請參閱???。

   下列範例事件模式只會傳送城市欄位中值為西雅圖的事件。

```
{
  "data": {
    "City": ["Seattle"]
  }
}
```

現在正在篩選事件，您可以為管道新增選擇性的擴充和目標。

定義事件擴充 (選用)

您可以將要擴充的事件資料傳送到 Lambda 函數、 AWS Step Functions 狀態機器、Amazon API Gateway 或 API 目的地。

若要選取擴充

1. 選擇擴充。
2. 在詳細資料下，針對服務，選取您要用於擴充的服務和相關設定。

您還可以在發送資料以進行增強之前對其進行轉換。

(選用) 定義輸入轉換器

1. 選擇擴充輸入轉換器 - 選用。
2. 針對範例事件/事件裝載，選擇範例事件類型。
3. 在轉換器中，輸入轉換器語法，例如，在 `"Event happened at <$.detail.field>."` 中，`<$.detail.field>` 是範例事件中欄位的參考。您也可以按兩下範例事件中的欄位，將其新增至轉換器。
4. 針對輸出，請確認輸出看起來像您想要的那樣。

現在資料已經過篩選和增強，您必須定義要將事件資料傳送至的目標。

設定目標

若要設定目標

1. 選擇 Target (目標)。
2. 在詳細資訊下，針對目標服務，選擇目標。顯示的欄位會隨您選擇的目標而異。請視需要輸入此目標類型的特定資訊。

您還可以在向目標發送資料之前對其進行轉換。

(選用) 定義輸入轉換器

1. 選擇目標輸入轉換器 - 選用。
2. 針對範例事件/事件裝載，選擇範例事件類型。

3. 在轉換器中，輸入轉換器語法，例如，在 "Event happened at <$.detail.field>." 中，<$.detail.field> 是範例事件中欄位的參考。您也可以按兩下範例事件中的欄位，將其新增至轉換器。
4. 針對輸出，請確認輸出看起來像您想要的那樣。

現在已規劃管道，請確保其設定已正確規劃。

規劃管道設定

依預設，管道處於作用中狀態，但您可以將其停用。您還可以指定管的權限、設置管道日誌和加入標籤。

規劃管道設定

1. 選擇管道設定標籤。
2. 根據預設，新建立的管道在建立後立即處於作用中狀態。如果要建立非作用中的管道，請在啟用下，針對啟用管道，關閉作用中。
3. 在權限下，針對執行角色，執行下列其中一項作業：
   a. 若要為此管道 EventBridge 建立新的執行角色，請選擇 [為此特定資源建立新角色]。在角色名稱底下，您可以選擇性地編輯角色名稱。
   b. 若要使用現有的角色，選擇使用現有的角色。在角色名稱下，選擇角色。
4. (選擇性) 如果您已指定 Kinesis 或 DynamoDB 串流作為管道來源，則可以設定重試原則和無效字母佇列 (DLQ)。

   針對重試政策和無效字母佇列 - 選用，請執行下列動作：

   在重試政策下，執行下列操作：

   a. 如果您想開啟重試政策，請開啟重試。根據預設，新建立的管道未開啟重試原則。
   b. 針對事件的最長存留期，輸入介於一分鐘 (00:01) 到 24 小時 (24:00) 之間的某個值。
   c. 針對重試嘗試，輸入介於 0 到 185 之間的某個數。
   d. 如果您想要使用無效字母佇列 (DLQ)，請開啟無效字母佇列，選擇您選擇的方法，然後選擇要使用的佇列或主題。根據預設，新建立的管道不使用 DLQ。
5. (選用) 在記錄 - 選用底下，您可以設定 EventBridge 管道如何將日誌資訊傳送至支援的服務，包括如何設定這些日誌。

如需記錄管道日誌的詳細資訊，請參閱 ???。

CloudWatch 依預設，會選取記錄檔做為記錄目的地，與記ERROR錄層級一樣。因此，依預設，P EventBridge ipes 會建立新的記 CloudWatch 錄群組，將包含詳細資料ERROR層級的記錄檔記錄傳送至該群組。

若要讓 P EventBridge ipes 將記錄檔記錄傳送至任何支援的記錄目的地，請執行下列動作：

a. 在日誌 - 選用底下，選擇要傳送日誌檔記錄的目的地。

b. 對於記錄層級，請選擇 EventBridge 要包含在記錄記錄中的資訊層級。預設會選取 ERROR 日誌層級。

   如需詳細資訊，請參閱 ???。

c. 如果您要 EventBridge 在記錄記錄中包含事件承載資訊、服務要求與回應資訊，請選取包含執行資料。

   如需詳細資訊，請參閱 ???。

d. 設定您選取的每個日誌目的地：

   對於 CloudWatch Logs 記錄檔，請在CloudWatch 記錄檔下執行下列動作：

   - 對於CloudWatch 記錄群組，請選擇是否要 EventBridge 建立新的記錄群組，或者您可以選取現有的記錄群組或指定現有記錄群組的 ARN。
   - 針對新的日誌群組，請視需要編輯日誌群組名稱。

   CloudWatch 預設會選取記錄檔。

   對於 Firehose 串流記錄，請在Firehose 串流記錄下選取 Firehose 串流。

   對於 Amazon S3 日誌，在 S3 日誌下執行以下操作：

   - 輸入儲存貯體的名稱來作為日誌目的地使用。
   - 輸入值區擁有者的 AWS 帳號 ID。
   - 輸入 EventBridge 建立 S3 物件時要使用的任何首碼文字。

     如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的利用首字組織物件。
   - 選擇您要格式化 S3 日誌記錄的 EventBridge 方式：

- `json`：JSON
- `plain`：純文字
- `w3c`：W3C 擴充日誌檔案格式

6. (選用) 在標籤 - 選用底下選擇新增標籤並為規則輸入一個或多個標籤。如需詳細資訊，請參閱 ???。
7. 選擇建立管道。

驗證組態參數

建立管路後， EventBridge 驗證下列組態參數：

- IAM 角色 — 由於管道建立後無法變更管道的來源，因此請 EventBridge 驗證提供的 IAM 角色是否可以存取來源。

> **Note**
>
> EventBridge 不會對擴充或目標執行相同的驗證，因為它們可以在管道建立之後進行更新。

- 批次處理 — EventBridge 驗證來源的批次大小不超過目標的最大批次大小。如果是這樣，則 EventBridge 需要較低的批次大小。此外，如果目標不支援批次處理，則無法在中設定來源的 EventBridge 批次處理。
- 擴充 — EventBridge 驗證 API Gateway 和 API 目的地擴充的批次大小是否為 1，因為只支援批次大小為 1。

啟動或停止管道

默認情況下，管道是 `Running` 並在創建時處理事件。

如果您使用 Amazon SQS、Kinesis 或 DynamoDB 來源建立管道，建立管道通常需要一兩分鐘的時間。

如果您使用 Amazon MSK、自我管理的 Apache Kafka 或 Amazon MQ 來源建立管道，則建立管道可能需要長達十分鐘的時間。

使用控制台建立管道而不處理事件

- 關閉啟用管道設定。

建立管道而不以程式設計方式處理事件

- 在您的 API 呼叫中，將 `DesiredState` 設定為 `Stopped`.

使用控制台啟動或停止既有管的步驟

- 在管道設定頁籤上的啟用下，對於啟用管道，打開或關閉作用中。

若要以程式設計方式啟動或停止既有管道

- 在 API 呼叫中，將 `DesiredState` 參數設定為 `RUNNING` 或 `STOPPED`。

管道是 `STOPPED` 和不再處理事件之間可能會有延遲：

- 對於 Amazon SQS 和串流來源，此延遲通常不到兩分鐘。
- 對於 Amazon MQ 和 Apache Kafka 來源而言，此延遲可能長達十五分鐘。

Amazon EventBridge 管道來源

EventBridge 管道會接收來自各種來源的事件資料，對該資料套用選用篩選器和擴充，然後將其傳送至目的地。

如果來源對傳送至 P EventBridge ipes 的事件強制執行順序，則該順序會在整個流程中保持到目的地。

可以將下列 AWS 服務指定為 EventBridge 管道的來源：

- Amazon DynamoDB stream
- Amazon Kinesis 串流
- Amazon MQ 代理程式
- Amazon MSK 串流
- Amazon SQS 佇列
- 阿帕奇卡夫卡流

  當您指定 Apache Kafka 串流作為管道來源時，您可以指定自己管理的 Apache Kafka 串流，或由第三方供應商管理的串流，例如：

  - Confluent Cloud

- CloudKarafka
- Redpanda

Amazon DynamoDB 串流作為來源

您可以使用 EventBridge 接管來接收 DynamoDB 串流中的記錄。然後，您可以選擇性地篩選或增強這些記錄，然後再將它們傳送到目標進行處理。您可以在設定管道時選擇 Amazon DynamoDB 串流的特定設定。當將資料傳送至目的地時，EventBridge 管道會維護資料串流中的記錄順序。

> **Important**
>
> 停用做為管道來源的 DynamoDB 串流會導致該管道變得無法使用，即使您之後重新啟用串流也是如此。發生這種情況的原因是：
>
> - 您無法停止、啟動或更新其來源已停用的管道。
> - 建立後，您無法使用新來源更新管道。當您重新啟用 DynamoDB 串流時，該串流會指派一個新的 Amazon Resource Name (ARN)，且不再與您的管道相關聯。
>
> 如果您確實重新啟用 DynamoDB 串流，則需要使用串流的新 ARN 建立新管道。

範例事件

下列範例事件顯示管道接收的資訊。您可以使用此事件來建立和篩選事件模式，或定義輸入轉換。並非所有欄位都可以篩選。如需有關篩選文字欄位的詳細資訊，請參閱 ???。

```
[
  {
    "eventID": "1",
    "eventVersion": "1.0",
    "dynamodb": {
      "Keys": {
        "Id": {
          "N": "101"
        }
      },
      "NewImage": {
        "Message": {
          "S": "New item!"
```

```
        },
        "Id": {
          "N": "101"
        }
      },
      "StreamViewType": "NEW_AND_OLD_IMAGES",
      "SequenceNumber": "111",
      "SizeBytes": 26
    },
    "awsRegion": "us-west-2",
    "eventName": "INSERT",
    "eventSourceARN": "arn:aws:dynamodb:us-east-1:111122223333:table/EventSourceTable",
    "eventSource": "aws:dynamodb"
  },
  {
    "eventID": "2",
    "eventVersion": "1.0",
    "dynamodb": {
      "OldImage": {
        "Message": {
          "S": "New item!"
        },
        "Id": {
          "N": "101"
        }
      },
      "SequenceNumber": "222",
      "Keys": {
        "Id": {
          "N": "101"
        }
      },
      "SizeBytes": 59,
      "NewImage": {
        "Message": {
          "S": "This item has changed"
        },
        "Id": {
          "N": "101"
        }
      },
      "StreamViewType": "NEW_AND_OLD_IMAGES"
    },
    "awsRegion": "us-west-2",
```

```
    "eventName": "MODIFY",
    "eventSourceARN": "arn:aws:dynamodb:us-east-1:111122223333:table/EventSourceTable",
    "eventSource": "aws:dynamodb"
  }
]
```

輪詢和批次處理串流

Lambda 會輪詢您 DynamoDB 串流中的碎片，其記錄的基本速率為每秒 4 次。當記錄可用時，EventBridge 會處理事件，並等待結果。如果處理成功， 會恢復輪詢，直到收到多筆記錄。

EventBridge 預設會在記錄可用時立即調用管道。如果 EventBridge 從來源中讀取的批次之中只有一筆記錄，只會處理一筆事件。為避免處理少量記錄，您可讓管道設定批次間隔，要求記錄緩衝記錄最長達五分鐘。處理事件之前，EventBridge 會繼續從事件來源中讀取記錄，直到收集到完整批次、批次間隔到期或者批次達到 6 MB 的承載限制。

您還可以透過並行處理來自每個碎片的多個批次來增加並行性。EventBridge 可同時處理每個碎片中最多 10 個批次。如果增加每個碎片的並行批次數量，EventBridge 仍會確保在分割區索引鍵層級進行依序處理。

規劃 `ParallelizationFactor` 設定來同時處理 Kinesis 或 DynamoDB 資料串流的一個碎片與多個管道執行。您可以透過從 1 (預設) 到 10 的並行化因子指定 EventBridge 從碎片輪詢的並行批次數。例如，當 `ParallelizationFactor` 設定為 2 時，您最多可以有 200 個並行 EventBridge 管道執行，來處理 100 個 Kinesis 資料碎片。當資料量急劇波動並且 `IteratorAge` 高時，這有助於縱向擴展處理輸送量。請注意，如果您使用 Kinesis 彙總，則並行化因子將不起作用。

輪詢和串流開始位置

請注意，建立和更新管道期間的串流輪詢最終會一致。

- 在建立管道期間，從串流開始輪詢事件可能需要幾分鐘時間。
- 在更新管道資源輪訓組態期間，從串流停止並重新開始輪詢事件可能需要幾分鐘時間。

這表示如果您指定 `LATEST` 當作串流的開始位置，管道可能會在建立或更新期間發送事件。若要確保沒有遺漏任何事件，請將串流開始位置指定為 `TRIM_HORIZON`。

報告批次項目失敗

EventBridge 取用和處理來源的串流資料時，依預設，只有在批次成功完成時，其檢查點才會到批次的最高序號。若要避免重新處理失敗批次中成功處理過的訊息，您可以設定擴充或目標，傳回哪些訊息成功，哪些失敗的物件。我們將其稱為部分批次回應。

如需更多詳細資訊，請參閱 ???。

成功與失敗條件

如果您傳回下列任一項目，EventBridge 會將批次視為完全成功：

- 空白 `batchItemFailure` 清單
- Null `batchItemFailure` 清單
- 空白 `EventResponse`
- Null `EventResponse`

如果您傳回下列任一項目，EventBridge 會將批次視為完全失敗：

- 空白字串 `itemIdentifier`
- Null `itemIdentifier`
- 具有錯誤金鑰名稱的 `itemIdentifier`

EventBridge 會根據您的重試政策來重試失敗。

Amazon Kinesis 串流做為來源

您可以使用 EventBridge 接管來接收 Kinesis 資料串流中的記錄。然後，您可以選擇性地篩選或增強這些記錄，然後再將它們傳送到可用的目的地之一進行處理。您可以在設定管道時選擇 Kinesis 特定的設定。當將資料傳送至目的地時，EventBridge 管道會維護資料串流中的記錄順序。

Kinesis 資料串流是一組碎片。每個碎片包含一系列的資料記錄。取用程式是處理來自 Kinesis 資料串流的資料的應用程式。您可以將 EventBridge 管道對應至共用輸送量取用程式 (標準迭代程式)，或對應至具有增強散發功能的專用輸送量取用程式。

對於標準迭代程式，EventBridge 會使用 HTTP 協定輪詢 Kinesis 串流中的每個碎片以尋找記錄。此函式會與碎片的其他消費者共用碎片讀取輸送量。

若要將延遲降至最低並最大化讀取輸送量，您可以建立具有增強散發功能的資料串流取用者。串流取用者會取得每個碎片的專用連線，其不會影響其他從串流讀取的應用程式。如果您有許多應用程式讀取相同的資料，或者，如果您要重新處理具有大量記錄的串流，則專屬的輸送量很有助益。Kinesis 透過 HTTP/2 推送記錄到 EventBridge。如需關於 Kinesis 資料串流的資訊，請參閱從 Amazon Kinesis Data Streams 讀取資料。

範例事件

下列範例事件顯示管道接收的資訊。您可以使用此事件來建立和篩選事件模式，或定義輸入轉換。並非所有欄位都可以篩選。如需有關所能篩選文字欄位的詳細資訊，請參閱 ???。


```
[
  {
    "kinesisSchemaVersion": "1.0",
    "partitionKey": "1",
    "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
    "data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
    "approximateArrivalTimestamp": 1545084650.987
    "eventSource": "aws:kinesis",
    "eventVersion": "1.0",
    "eventID":
 "shardId-000000000006:49590338271490256608559692538361571095921575989136588898",
    "eventName": "aws:kinesis:record",
    "invokeIdentityArn": "arn:aws:iam::123456789012:role/lambda-role",
    "awsRegion": "us-east-2",
    "eventSourceARN": "arn:aws:kinesis:us-east-2:123456789012:stream/lambda-stream"
  },
  {
    "kinesisSchemaVersion": "1.0",
    "partitionKey": "1",
    "sequenceNumber": "49590338271490256608559692540925702759324208523137515618",
    "data": "VGhpcyBpcyBvbmx5IGEgdGVzdC4=",
    "approximateArrivalTimestamp": 1545084711.166
    "eventSource": "aws:kinesis",
    "eventVersion": "1.0",
    "eventID":
 "shardId-000000000006:49590338271490256608559692540925702759324208523137515618",
    "eventName": "aws:kinesis:record",
    "invokeIdentityArn": "arn:aws:iam::123456789012:role/lambda-role",
    "awsRegion": "us-east-2",
    "eventSourceARN": "arn:aws:kinesis:us-east-2:123456789012:stream/lambda-stream"
  }
```

```
]
```

輪詢和批次處理串流

EventBridge 會輪詢您 Kinesis 串流中的碎片，其記錄的基本速率為每秒 4 次。當記錄可用時，EventBridge 會處理事件，並等待結果。如果處理成功，EventBridge 會恢復輪詢，直到收到多筆記錄。

EventBridge 預設會在記錄可用時立即調用管道。如果 EventBridge 從來源中讀取的批次之中只有一筆記錄，只會處理一筆事件。為避免處理具有少量記錄的函數，您可讓管道設定批次間隔，要求管道緩衝記錄最長達五分鐘。處理事件之前，EventBridge 會繼續從來源中讀取記錄，直到收集到完整批次、批次間隔到期或者批次達到 6 MB 的承載限制。

您還可以透過並行處理來自每個碎片的多個批次來增加並行性。EventBridge 可同時處理每個碎片中最多 10 個批次。如果增加每個碎片的並行批次數量，EventBridge 仍會確保在分割區索引鍵層級進行依序處理。

規劃 `ParallelizationFactor` 設定來同時處理 Kinesis 或 DynamoDB 資料串流的一個碎片與多個管道執行。您可以透過從 1 (預設) 到 10 的並行化因子指定 EventBridge 從碎片輪詢的並行批次數。例如，當 `ParallelizationFactor` 設定為 2 時，您最多可以有 200 個並行 EventBridge 管道執行，來處理 100 個 Kinesis 資料碎片。當資料量急劇波動並且 `IteratorAge` 高時，這有助於擴展處理輸送量。請注意，如果您使用 Kinesis 彙總，則並行化因子將不起作用。

輪詢和串流開始位置

請注意，建立和更新管道期間的串流輪詢最終會一致。

- 在建立管道期間，從串流開始輪詢事件可能需要幾分鐘時間。
- 在更新管道期間，從串流停止並重新開始輪詢設定可能需要幾分鐘時間。

這表示如果您指定 `LATEST` 當作串流的開始位置，管道可能會在建立或更新期間遺漏事件。若要確保沒有遺漏任何事件，請將串流開始位置指定為 `TRIM_HORIZON` 或 `AT_TIMESTAMP`。

報告批次項目失敗

EventBridge 取用和處理來源的串流資料時，依預設，只有在批次成功完成時，其檢查點才會到批次的最高序號。若要避免重新處理失敗批次中成功處理過的訊息，您可以設定擴充或目標，傳回哪些訊息成功，哪些失敗的物件。我們將其稱為部分批次回應。

如需更多詳細資訊，請參閱 ???。

成功與失敗條件

如果您傳回下列任一項目，EventBridge 會將批次視為完全成功：

- 空白 `batchItemFailure` 清單
- Null `batchItemFailure` 清單
- 空白 `EventResponse`
- Null `EventResponse`

如果您傳回下列任一項目，EventBridge 會將批次視為完全失敗：

- 空白字串 `itemIdentifier`
- Null `itemIdentifier`
- 具有錯誤金鑰名稱的 `itemIdentifier`

EventBridge 會根據您的重試政策來重試失敗。

Amazon MQ 訊息代理程式做為來源

您可以用 EventBridge 接收來自 Amazon MQ 訊息代理程式的記錄。然後，您可以選擇性地篩選或增強這些記錄，然後再將它們傳送到可用的目的地之一進行處理。您可以在設定管道時選擇 Amazon MQ 特定的設定。當將資料傳送至目的地時，EventBridge 管道會維護來自訊息代理程式的資料串流中的記錄順序。

Amazon MQ 是一項受管訊息代理程式服務，適用於 Apache ActiveMQ 和 RabbitMQ。訊息代理程式透過主題或佇列事件目的地，允許軟體應用程式和元件使用不同程式設計語言、作業系統和正式簡訊協定來進行通訊。

Amazon MQ 還可以透過安裝 ActiveMQ 或 RabbitMQ 代理程式，來代表您管理 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。安裝代理程式之後，它會為您的執行個體提供不同的網路拓撲和其他基礎結構需求。

Amazon MQ 來源具有下列組態限制：

- 跨帳戶：EventBridge 不支援跨帳戶處理。您無法用 EventBridge 來處理來自不同 AWS 帳戶中的 Amazon MQ 訊息代理程式的記錄。

- 身分驗證支援：對於 ActiveMQ，僅支援 ActiveMQ SimpleAuthenticationPlugin。對於 RabbitMQ，僅支援 PLAIN 身分驗證機制。若要管理認證，請使用 AWS Secrets Manager。如需 ActiveMQ 身分驗證的詳細資訊，請參閱 Amazon MQ 開發人員指南中的將 ActiveMQ 代理程式與 LDAP 整合。
- 連線配額：代理程式每個線路層級協定允許的連線數量上限。此配額以代理程式執行個體類型為基礎。如需詳細資訊，請參閱 Amazon MQ 開發人員指南中的 Amazon MQ 中的配額的代理程式。
- 連線：您可以在公有或私有虛擬私有雲端 (VPC) 中建立代理程式。若是私有 VPC，您的管道需要存取 VPC 才能接收訊息。
- 事件目的地：僅支援佇列目的地。然而，您可以使用虛擬主題，當作為佇列在外部與管道互動時，其行為在內部可作為主題。如需詳細資訊，請參閱 Apache ActiveMQ 網站上的虛擬目的地，以及 RabbitMQ 網站上的虛擬主機。
- 網路拓撲：對於 ActiveMQ，管道僅支援單一執行個體或待命代理程式。對於 RabbitMQ，每個管道僅支援單一執行個體代理程式或叢集部署。單一執行個體代理程式需要容錯移轉端點。如需這些代理程式部署模式的詳細資訊，請參閱 Amazon MQ 開發人員指南中的 ActiveMQ 代理程式架構和 Rabbit MQ 代理程式架構。
- 協定：支援的協定取決於您所使用的 Amazon MQ。
  - 對於 ActiveMQ 整合，EventBridge 會使用 OpenWire/Java 訊息服務 (JMS) 協定來取用訊息。任何其他通訊協定都不支援訊息消耗。EventBridge 僅支援 JMS 通訊協定中的文字訊息和位元組輸入作業。如需有關 OpenWire 協定的詳細資訊，請參閱 Apache ActiveMQ 網站上的 OpenWire。
  - 對於 RabbitMQ 整合，EventBridge 透過 AMQP 0-9-1 協定來取用訊息。不支援透過其他協定來取用訊息。如需 RabbitMQ 實作 AMQP 0-9-1 協定的詳細資訊，請參閱 RabbitMQ 網站上的 AMQP 0-9-1 完整參考指南。

EventBridge 會自動支援 Amazon MQ 支援的最新版 ActiveMQ 和 RabbitMQ。如需支援的最新版，請參閱Amazon MQ 開發人員指南中的 Amazon MQ 版本備註。

> **Note**
>
> 根據預設，Amazon MQ 具有每週代理程式維護時段。代理程式在該時段不可用。對於沒有待命的代理程式，EventBridge 不會處理訊息，直到視窗結束為止。

範例事件

下列範例事件顯示管道接收的資訊。您可以使用此事件來建立和篩選事件模式，或定義輸入轉換。並非所有欄位都可以篩選。如需有關所能篩選文字欄位的詳細資訊，請參閱 ???。

ActiveMQ

```
[
  {
    "eventSource": "aws:amq",
    "eventSourceArn": "arn:aws:mq:us-
west-2:112556298976:broker:test:b-9bcfa592-423a-4942-879d-eb284b418fc8",
    "messageID": "ID:b-9bcfa592-423a-4942-879d-eb284b418fc8-1.mq.us-
west-2.amazonaws.com-37557-1234520418293-4:1:1:1:1",
    "messageType": "jms/text-message",
    "data": "QUJDOkFBQUE=",
    "connectionId": "myJMSCoID",
    "redelivered": false,
    "destination": {
      "physicalname": "testQueue"
    },
    "timestamp": 1598827811958,
    "brokerInTime": 1598827811958,
    "brokerOutTime": 1598827811959
  },
  {
    "eventSource": "aws:amq",
    "eventSourceArn": "arn:aws:mq:us-
west-2:112556298976:broker:test:b-9bcfa592-423a-4942-879d-eb284b418fc8",
    "messageID": "ID:b-9bcfa592-423a-4942-879d-eb284b418fc8-1.mq.us-
west-2.amazonaws.com-37557-1234520418293-4:1:1:1:1",
    "messageType": "jms/bytes-message",
    "data": "3DTOOW7crj51prgVLQaGQ82S48k=",
    "connectionId": "myJMSCoID1",
    "persistent": false,
    "destination": {
      "physicalname": "testQueue"
    },
    "timestamp": 1598827811958,
    "brokerInTime": 1598827811958,
    "brokerOutTime": 1598827811959
  }
]
```

RabbitMQ

```
[
  {
```

```
    "eventSource": "aws:rmq",
    "eventSourceArn": "arn:aws:mq:us-
west-2:111122223333:broker:pizzaBroker:b-9bcfa592-423a-4942-879d-eb284b418fc8",
    "eventSourceKey": "pizzaQueue::/",
    "basicProperties": {
      "contentType": "text/plain",
      "contentEncoding": null,
      "headers": {
        "header1": {
          "bytes": [
            118,
            97,
            108,
            117,
            101,
            49
          ]
        },
        "header2": {
          "bytes": [
            118,
            97,
            108,
            117,
            101,
            50
          ]
        },
        "numberInHeader": 10
      },
      "deliveryMode": 1,
      "priority": 34,
      "correlationId": null,
      "replyTo": null,
      "expiration": "60000",
      "messageId": null,
      "timestamp": "Jan 1, 1970, 12:33:41 AM",
      "type": null,
      "userId": "AIDACKCEVSQ6C2EXAMPLE",
      "appId": null,
      "clusterId": null,
      "bodySize": 80
    },
    "redelivered": false,
```

```
    "data": "eyJ0aW1lb3V0IjowLCJkYXRhIjoiQ1pybWYwR3c4T3Y0YnFMUXhENEUifQ=="
  }
]
```

取用者群組

若要與 Amazon MQ 互動，EventBridge 會建立可以從您的 Amazon MQ 代理程式讀取的取用者群體。建立取用者群組時，會使用與管道 UUID 相同的 ID。

對於 Amazon MQ 來源，EventBridge 會批次處理記錄，並在單個承載中將它們傳送到您的函數。要控制行為，您可以設定批次間隔和批次大小。EventBridge 會提取訊息，直到發生下列一種情況：

- 處理過的記錄達到有效負載大小最大 6 MB。
- 批次化視窗即會到期。
- 記錄數達到完整批次大小。

EventBridge 將您的批次轉換為單個承載，然後調用您的函數。訊息不會保存也不會還原序列化。相反，取用者群組會將其擷取為位元組 BLOB。然後，透過 base64 將它們編碼到 JSON 承載中。如果管道針對批次中的任何訊息傳回錯誤，EventBridge 會重試整個批次的訊息，直至處理成功或訊息過期。

網路組態

根據預設，Amazon MQ 代理程式使用設定為 false 的 `PubliclyAccessible` 標記建立。只有當 `PubliclyAccessible` 設定為 true 時，代理程式才會取得公有 IP 地址。針對使用管道的完整存取權，您的代理程式必須使用公有端點，或提供對 VPC 的存取權。

如果您無法公開存取您的 Amazon MQ 代理程式，EventBridge 必須能存取與代理程式相關聯的 Amazon Virtual Private Cloud (Amazon VPC)。若要存取 Amazon MQ 代理程式的 VPC，EventBridge 需要對來源子網路進行輸出網際網路存取。對於公用子網路，這必須是受管理的 NAT 閘道。對於私有子網路，它可以是 NAT 閘道，也可以是您自己的 NAT。確定 NAT 具有公有 IP 地址，可連線至網際網路。

使用下列規則 (最低限度) 設定 Amazon VPC 安全群組：

- 傳入規則：針對沒有公開可存取性的代理程式，允許指定為您的來源的安全群組所有連接埠上的所有流量。針對具有公開可存取性的代理程式，允許所有目的地所有連接埠上的所有流量。
- 傳出規則：允許所有目的地的所有連接埠上的所有流量。

> **Note**
>
> 您的 Amazon VPC 組態可透過 Amazon MQ API 來探索。您不需要在設定期間來設定它。

Amazon Managed Streaming for Apache Kafka 主題作為來源

您可以使用 EventBridge 管道從 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 主題中接收記錄。然後，您可以選擇性地篩選或增強這些記錄，然後再將它們傳送到可用的目的地之一進行處理。您可以在設定管道時選擇 Amazon MSK 特定的設定。當將資料傳送至目的地時，EventBridge 管道會維護來自訊息代理程式的資料串流中的記錄順序。

Amazon MSK 是一項全受管服務，可讓您建立和執行使用 Apache Kafka 處理串流資料的應用程式。Amazon MSK 可簡化執行 Kafka 叢集的設定、擴展和管理。Amazon MSK 也可讓您更輕鬆地為應用程式設定多個可用區域，並透過 AWS Identity and Access Management (IAM) 實現安全。Amazon MSK 支援 Kafka 的多個開放原始碼版本。

Amazon MSK 作為來源時，其運作方式類似於使用 Amazon Simple Queue Service (Amazon SQS) 或 Amazon Kinesis。EventBridge 會在內部輪詢來源中的新訊息，然後同步調用目標。EventBridge 會批次讀取訊息，並將這些訊息作為事件裝載提供給函數。批次大小上限可設定。(預設值為 100 則訊息。)

對於基於 Apache Kafka 的來源，EventBridge 支援處理控制參數，例如批次間隔和批次大小。

EventBridge 會依序讀取每個分割區的訊息。EventBridge 處理每個批次後，會遞交該批次中訊息的偏移量。如果管道目標針對批次中的任何訊息傳回錯誤，EventBridge 會重試整個批次的訊息，直至處理成功或訊息過期。

EventBridge 會在調用您的目標時在事件中傳送訊息批次。事件裝載包含訊息陣列。陣列中的每個項目包含 Amazon MSK 主題和分割區識別符的詳細資訊，以及時間戳記和 base64 編碼的訊息。

範例事件

下列範例事件顯示管道接收的資訊。您可以使用此事件來建立和篩選事件模式，或定義輸入轉換。並非所有欄位都可以篩選。如需有關所能篩選欄位的詳細資訊，請參閱 ???。

```
[
  {
    "eventSource": "aws:kafka",
    "eventSourceArn": "arn:aws:kafka:sa-east-1:123456789012:cluster/
vpc-2priv-2pub/751d2973-a626-431c-9d4e-d7975eb44dd7-2",
```

```
    "eventSourceKey": "mytopic-0",
    "topic": "mytopic",
    "partition": "0",
    "offset": 15,
    "timestamp": 1545084650987,
    "timestampType": "CREATE_TIME",
    "key":"abcDEFghiJKLmnoPQRstuVWXyz1234==",
    "value":"SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
    "headers": [
      {
        "headerKey": [
          104,
          101,
          97,
          100,
          101,
          114,
          86,
          97,
          108,
          117,
          101
        ]
      }
    ]
  }
]
```

輪詢和串流開始位置

請注意，建立和更新管道期間的串流輪詢最終會一致。

- 在建立管道期間，從串流開始輪詢事件可能需要幾分鐘時間。
- 在更新管道期間，從串流停止並重新開始輪詢設定可能需要幾分鐘時間。

這表示如果您指定 LATEST 當作串流的開始位置，管道可能會在建立或更新期間遺漏事件。若要確保沒有遺漏任何事件，請將串流開始位置指定為 TRIM_HORIZON。

MSK 叢集身分驗證

EventBridge 需要許可才能存取 Amazon MSK 叢集、擷取記錄和執行其他任務。Amazon MSK 支援數個選項，用於控制用戶端對 MSK 叢集的存取。如需此身分驗證方法的詳細資訊，請參閱 ???。

叢集存取選項

- 未驗證的存取
- SASL/SCRAM 身分驗證
- IAM 角色型身分驗證
- 交互 TLS 驗證
- 設定 mTLS 機密
- EventBridge 選擇引導代理程式的方法

未驗證的存取

我們建議您只使用未經驗證的存取進行開發。只有在叢集停用 IAM 角色型驗證時，未驗證的存取才能運作。

SASL/SCRAM 身分驗證

Amazon MSK 支援 Simple Authentication and Security Layer/Salted Challenge Response Authentication Mechanism (SASL/SCRAM) 身分驗證與 Transport Layer Security (TLS) 加密。若要讓 EventBridge 連線至叢集，您可以將身分驗證憑證 (登入憑證) 存放在 AWS Secrets Manager 機密中。

如需使用 Secrets Manager 詳細資訊，請參閱《Amazon Managed Streaming for Apache Kafka 開發人員指南》中的AWS Secrets Manager 的使用者名稱和密碼身分驗證。

Amazon MSK 不支援 SASL/PLAIN 身分驗證。

IAM 角色型身分驗證

您可以使用 IAM 來驗證連至 MSK 叢集之用戶端的身分。若您 MSK 叢集上的 IAM 身分驗證為作用中，且您未提供身分驗證密碼，則 EventBridge 會自動預設使用 IAM 身分驗證。若要建立和部署 IAM 使用者或角色型政策，請使用 IAM 主控台或 API。如需詳細資訊，請參閱《Amazon Managed Streaming for Apache Kafka 開發人員指南》中的IAM 存取控制。

若要允許 EventBridge 連線至 MSK 叢集、讀取記錄，以及執行其他必要動作，請將下列許可新增至函數的執行角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:DescribeGroup",
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeTopic",
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeClusterDynamicConfiguration"
            ],
            "Resource": [
                "arn:aws:kafka:region:account-id:cluster/cluster-name/cluster-uuid",
                "arn:aws:kafka:region:account-id:topic/cluster-name/cluster-uuid/topic-
name",
                "arn:aws:kafka:region:account-id:group/cluster-name/cluster-
uuid/consumer-group-id"
            ]
        }
    ]
}
```

您可以將這些許可的範圍設定為特定叢集、主題和群組。如需詳細資訊，請參閱《Amazon Managed Streaming for Apache Kafka 開發人員指南》中的Amazon MSK Kafka 動作。

交互 TLS 驗證

相互 TLS (mTLS) 可提供用戶端與伺服器之間的雙向身分驗證。用戶端會將憑證傳送至伺服器以供伺服器驗證用戶端，而伺服器會將憑證傳送至用戶端以供用戶端驗證伺服器。

若為 Amazon MSK，EventBridge 會以用戶端的身分運作。您可以設定用戶端憑證 (做為 Secrets Manager 中的機密) 來驗證 EventBridge 與 MSK 叢集中的代理程式。客户憑證必須由伺服器信任存放區中的憑證授權機構 (CA) 簽署。MSK 叢集會傳送伺服器憑證到 EventBridge，並用 EventBridge 來驗證代理程式。伺服器端憑證必須由 AWS 信任存放區中的憑證授權機構簽署。

Amazon MSK 不支援自行簽署的伺服器憑證，因為 Amazon MSK 中的所有代理程式都使用由 Amazon Trust Services CAs (根據預設，EventBridge 信任此機構) 簽署的公有憑證。

如需有關適用於 Amazon MSK 的 mTLS 之詳細資訊，請參閱《Amazon Managed Streaming for Apache Kafka 開發人員指南》中的相互 TLS 身分驗證。

設定 mTLS 機密

CLIENT_CERTIFICATE_TLS_AUTH 機密必須有憑證欄位和私有金鑰欄位。若為加密的私有金鑰，機密需要私有金鑰密碼。憑證與私有金鑰均必須為 PEM 格式。

> **Note**
>
> EventBridge 管道支援 PBES1 (但不支援 PBES2) 私有金鑰加密演算法。

憑證欄位必須包含憑證清單，以用戶端憑證開頭，隨後則是任何中繼憑證，並以根憑證結尾。每個憑證均必須以新的一行開始，結構如下：

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager 支援高達 65,536 個位元組的機密，此空間足以容納長憑證鏈。

私有金鑰必須為 PKCS #8 格式，結構如下：

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

對於已加密的私有金鑰，請使用下列結構：

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

下列範例顯示的是使用了已加密私有金鑰之 mTLS 身分驗證的機密內容。若為加密的私有金鑰，您可以在機密中包含私有金鑰密碼。

```
{
 "privateKeyPassword": "testpassword",
 "certificate": "-----BEGIN CERTIFICATE-----
MIIE5DCCAsygAwIBAgIRAPJdwaFaNRrytHBto0j5BA0wDQYJKoZIhvcNAQELBQAw
...
j0Lh4/+1HfgyE2KlmII36dg4IMzNjAFEBZiCRoPimO40s1cRqtFHXoal0QQbIlxk
cmUuiAii9R0=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFgjCCA2qgAwIBAgIQdjNZd6uFf9hbNC5RdfmHrzANBgkqhkiG9w0BAQsFADBb
...
rQoiowbbk5wXCheYSANQIfTZ6weQTgiCHCCbuuMKNVS95FkXm0vqVD/YpXKwA/no
```

```
c8PH3PSoAaRwMMgOSA2ALJvbRz8mpg==
-----END CERTIFICATE-----",
 "privateKey": "-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFKzBVBgkqhkiG9w0BBQ0wSDAnBgkqhkiG9w0BBQwwGgQUiAFcK5hT/X7Kjmgp
...
QrSekqF+kWzmB6nAfSzgO9IaoAaytLvNgGTckWeUkWn/V0Ck+LdGUXzAC4RxZnoQ
zp2mwJn2NYB7AZ7+imp0azDZb+8YG2aUCiyqb6PnnA==
-----END ENCRYPTED PRIVATE KEY-----"
}
```

EventBridge 選擇引導代理程式的方法

EventBridge 會依據叢集上可用的身分驗證方法，以及您是否提供了身分驗證密碼，以此選擇引導代理程式。若您提供了 MTL 或 SASL/SCRAM 的密碼，則 EventBridge 會自動選擇該身分驗證方法。若您未提供密碼，EventBridge 會選取叢集上作用中的安全強度最高的身分驗證方法。以下是 EventBridge 選擇代理程式的優先順序，身分驗證安全強度依次遞減：

- mTLS (已提供 mTLS 密碼)
- SASL/SCRAM (已提供 SASL /SCROM 密碼)
- SASL IAM (未提供任何密碼，且 IAM 身分驗證在作用中)
- 未驗證的 TLS (未提供任何密碼，且 IAM 身分驗證未在作用中)
- 純文字 (未提供任何密碼，且 IAM 身分驗證和未經身分驗證的 TLS 皆未在作用中)

> **Note**
>
> 若 EventBridge 無法連線至最安全的代理程式類型， 便不會嘗試連線至其他 (安全強度較弱) 的代理程式類型。若您要讓 EventBridge 選擇安全強度較弱較弱的代理程式類型，請停用叢集上所有安全強度較弱更高的身分驗證方法。

網路組態

EventBridge 必須能存取與您 Amazon MSK 叢集相關聯的 Amazon Virtual Private Cloud (Amazon VPC) 資源。若要存取 Amazon MSK 叢集的 VPC 人雲端，EventBridge 需要對來源子網路進行輸出網際網路存取。對於公用子網路，這必須是受管理的 NAT 閘道。對於私有子網路，它可以是 NAT 閘道，也可以是您自己的 NAT。確定 NAT 具有公有 IP 地址，可連線至網際網路。

使用下列規則 (最低限度) 設定 Amazon VPC 安全群組：

- 傳入規則：針對來源指定的安全群組，允許 Amazon MSK 代理程式連接埠上的所有流量 (9092 代表純文字、9094 代表 TLS、9096 代表 SASL、9098 代表 IAM)。
- 傳出規則：針對所有目的地，允許連接埠 443 上的所有流量。針對來源指定的安全群組，允許 Amazon MSK 代理程式連接埠上的所有流量 (9092 代表純文字、9094 代表 TLS、9096 代表 SASL、9098 代表 IAM)。

> **Note**
>
> 您的 Amazon VPC 組態可透過 Amazon MSK API 來探索。您不需要在設定期間設定它。

可自訂的取用者群組 ID

將 Kafka 設為來源時，您可以指定取用者群組 ID。此取用者群組 ID 是您希望管道加入之 Kafka 取用者群組的現有識別符。您可以使用此特徵將任何進行中的 Kafka 記錄處理設定從其他取用者無縫遷移至 EventBridge。

如果您指定取用者群組 ID，且該取用者群組內還有其他作用中的輪詢者，則 Kafka 會將訊息分配給所有取用者。換句話說，EventBridge 不會收到有關 Kafka 主題的所有訊息。如果您希望 EventBridge 處理主題中的所有訊息，請關閉該取用者群組中的任何其他輪詢者。

此外，如果您指定取用者群組 ID，且 Kafka 找到具有相同 ID 的有效現有取用者群組，則 EventBridge 會忽略用於管道的 `StartingPosition` 參數。相反的，EventBridge 會根據取用者群組的承諾偏移量開始處理記錄。如果您指定取用者群組 ID，但 Kafka 找不到現有的取用者群組，則 EventBridge 會使用指定的 `StartingPosition` 來設定來源。

您指定的取用者群組 ID 在所有 Kafka 事件來源中必須是唯一的。使用指定的取用者群組 ID 建立管道之後，您就無法更新此值。

Amazon MSK 事件來源的自動擴展

當您最初建立 Amazon MSK 來源時，EventBridge 會分配一個取用者來處理 Kafka 主題的所有分割區。每個取用者都有多個並行運行的處理器以處理增加的工作負載。此外，會根據工作負載自動增加或減少取用者數量。為了保留每個分割區中的訊息順序，主題中每個分割區的取用者數上限是一個取用者。

每隔 1 分鐘，EventBridge 會評估主題中所有分割區的取用者偏移延遲。如果延遲太高，則表示分割區接收訊息的速度比 EventBridge 處理訊息的速度更快。如有必要，EventBridge 會新增或移除主題取用者。新增或刪除取用者的擴展過程，將在三分鐘的評估期間內完成。

如果您的目標過載，則 EventBridge 會減少取用者的數量。此動作可透過減少取用者可擷取和傳送至管道的訊息數量，減少管道的工作負載。

阿帕奇卡夫卡流作為源

Apache Kafka 是開源事件串流平台，可支援資料管道和串流分析等工作負載。您可以使用 Amazon Managed Streaming for Apache Kafka（Amazon MSK），或自我管理的阿帕奇卡夫卡集群。在 AWS 術語中，自我管理叢集是指任何不由主控的 Apache Kafka 叢集。 AWS這包括您自己管理的叢集，以及由第三方提供者託管的叢集 Confluent Cloud，例如CloudKarafka、或Redpanda。

如需叢集其他 AWS 託管選項的詳細資訊，請參閱AWS 大數據部落格AWS上的執行 Apache Kafka 的最佳實務。

阿帕奇卡夫卡作為源的操作類似於使用 Amazon Simple Queue Service (Amazon SQS) 或 Amazon Kinesis。 EventBridge內部輪詢來自來源的新郵件，然後同步調用目標。 EventBridge 批量讀取消息，並將這些消息作為事件有效負載提供給您的函數。批次大小上限可設定。(預設值為 100 則訊息。)

對於以 Apache Kafka 為基礎的來源， EventBridge 支援處理控制參數，例如批次處理視窗和批次大小。

EventBridge 當它調用管道時，在事件參數中發送一批消息。事件裝載包含訊息陣列。陣列中的每個項目包含 Apache Kafka 主題和 Kafka 分割區識別符的詳細資訊，以及時間戳記和 base64 編碼的訊息。

範例事件

下列範例事件顯示管道接收的資訊。您可以使用此事件來建立和篩選事件模式，或定義輸入轉換。並非所有欄位都可以篩選。如需有關所能篩選欄位的詳細資訊，請參閱 ???。

```
[
  {
    "eventSource": "SelfManagedKafka",
    "bootstrapServers": "b-2.demo-cluster-1.a1bcde.c1.kafka.us-
east-1.amazonaws.com:9092,b-1.demo-cluster-1.a1bcde.c1.kafka.us-
east-1.amazonaws.com:9092",
    "eventSourceKey": "mytopic-0",
    "topic": "mytopic",
    "partition": 0,
    "offset": 15,
    "timestamp": 1545084650987,
    "timestampType": "CREATE_TIME",
    "key":"abcDEFghiJKLmnoPQRstuVWXyz1234==",
    "value":"SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
```

```
    "headers": [
      {
        "headerKey": [
          104,
          101,
          97,
          100,
          101,
          114,
          86,
          97,
          108,
          117,
          101
        ]
      }
    ]
  }
]
```

Apache Kafka 叢集身分驗證

EventBridge 管道支持多種方法來與您的自我管理的 Apache 卡夫卡集群進行身份驗證。請確保您已設定 Apache Kafka 叢集使用其中一種支援的身分驗證方法。如需有關 Apache Kafka 安全性的詳細資訊，請參閱 Kafka 文件的安全性區段。

VPC 存取

如果您使用的是自我管理的 Apache Kafka 環境，只有您的 VPC 中的 Apache 卡夫卡使用者可以存取您的 Apache 卡夫卡代理程式，您必須在 Apache 卡夫卡來源中設定 Amazon Virtual Private Cloud (Amazon VPC)。

SASL/SCRAM 身分驗證

EventBridge 管道支援具有傳輸層安全性 (TLS) 加密的簡易驗證和安全層/加密挑戰回應驗證機制 (SASL/SCRAM) 驗證。 EventBridge 管道會傳送加密的認證，以便與叢集進行驗證。如需 SASL/SCRAM 身分驗證的詳細資訊，請參閱 RFC 5802。

EventBridge 管道支援使用 TLS 加密的 SASL/一般驗證。使用 SASL/PLAIN 驗證時， EventBridge 管道會將認證以純文字 (未加密) 的形式傳送至伺服器。

若使用 SASL 身分驗證，您需將登入憑證儲存為 AWS Secrets Manager中的秘密。

交互 TLS 驗證

相互 TLS (mTLS) 可提供用戶端與伺服器之間的雙向身分驗證。用戶端會將憑證傳送至伺服器以供伺服器驗證用戶端，而伺服器會將憑證傳送至用戶端以供用戶端驗證伺服器。

在自我管理的阿帕奇卡夫卡， EventBridge 管道充當客戶端。您可以設定用戶端憑證 (做為秘密管理員中的秘密)，以使用 Apache Kafka 代理程式驗證 EventBridge 管道。客户憑證必須由伺服器信任存放區中的憑證授權機構 (CA) 簽署。

阿帕奇卡夫卡集群發送服務器證書管道與 EventBridge 管道來驗證 Apache 卡夫卡經紀人。EventBridge 伺服器憑證可以是公有憑證授權機構憑證或私有憑證授權機構/自行簽署的憑證。公用 CA 憑證必須由位於 P EventBridge ipes 信任存放區中的 CA 簽署。對於私人 CA /自我簽署證書，您可以配置服務器根 CA 證書（作為秘 Secrets Manager 器中的密碼）。 EventBridge 管道使用根證書來驗證阿帕奇卡夫卡經紀人。

如需有關 mTLS 的詳細資訊，請參閱將 Amazon MSK 的相互 TLS 身分驗證作為來源。

設定用戶端憑證機密

CLIENT_CERTIFICATE_TLS_AUTH 機密必須有憑證欄位和私有金鑰欄位。若為加密的私有金鑰，機密需要私有金鑰密碼。憑證與私有金鑰均必須為 PEM 格式。

> **Note**
>
> EventBridge 管道支援 PBES1 (但不支援 PBES2) 私密金鑰加密演算法。

憑證欄位必須包含憑證清單，以用戶端憑證開頭，隨後則是任何中繼憑證，並以根憑證結尾。每個憑證均必須以新的一行開始，結構如下：

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager 支援高達 65,536 個位元組的機密，此空間足以容納長憑證鏈。

私有金鑰必須為 PKCS #8 格式，結構如下：

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

對於已加密的私有金鑰，請使用下列結構：

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

下列範例顯示的是使用了已加密私有金鑰之 mTLS 身分驗證的機密內容。若為加密的私有金鑰，請在機密中包含私有金鑰密碼。

```
{
 "privateKeyPassword": "testpassword",
 "certificate": "-----BEGIN CERTIFICATE-----
MIIE5DCCAsygAwIBAgIRAPJdwaFaNRrytHBto0j5BA0wDQYJKoZIhvcNAQELBQAw
...
j0Lh4/+1HfgyE2KlmII36dg4IMzNjAFEBZiCRoPimO40s1cRqtFHXoal0QQbIlxk
cmUuiAii9R0=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFgjCCA2qgAwIBAgIQdjNZd6uFf9hbNC5RdfmHrzANBgkqhkiG9w0BAQsFADBb
...
rQoiowbbk5wXCheYSANQIfTZ6weQTgiCHCCbuuMKNVS95FkXm0vqVD/YpXKwA/no
c8PH3PSoAaRwMMgOSA2ALJvbRz8mpg==
-----END CERTIFICATE-----",
 "privateKey": "-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFKzBVBgkqhkiG9w0BBQ0wSDAnBgkqhkiG9w0BBQwwGgQUiAFcK5hT/X7Kjmgp
...
QrSekqF+kWzmB6nAfSzgO9IaoAaytLvNgGTckWeUkWn/V0Ck+LdGUXzAC4RxZnoQ
zp2mwJn2NYB7AZ7+imp0azDZb+8YG2aUCiyqb6PnnA==
-----END ENCRYPTED PRIVATE KEY-----"
}
```

設定伺服器根憑證授權機構憑證機密

如果您的 Kafka 代理程式使用 TLS 加密與私有憑證授權機構簽署的憑證，則您建立此機密。您可以使用 TLS 加密以供 VPC、SASL/SCRAM、SASL/PLAIN 或 mTLS 身分驗證之用。

伺服器根 CA 憑證密碼必須有包含 PEM 格式的 Apache Kafka 代理程式根 CA 憑證的欄位。下列範例說明機密的結構。

```
{
     "certificate": "-----BEGIN CERTIFICATE-----
  MIID7zCCAtegAwIBAgIBADANBgkqhkiG9w0BAQsFADCBmDELMAkGA1UEBhMCVVMx
```

```
EDAOBgNVBAgTB0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxJTAjBgNVBAoT
HFN0YXJmaWVsZCBUZWNobm9sb2dpZXMsIEluYy4xOzA5BgNVBAMTMlN0YXJmaWVs
ZCBTZXJ2aWNlcyBSb290IENlcnRpZmljYXRlIEF1dG...
-----END CERTIFICATE-----"
```

網路組態

如果您使用的是使用私有 VPC 連線的自我管理 Apache Kafka 環境，則 EventBridge 必須能夠存取與 Apache Kafka 代理程式相關聯的 Amazon Virtual Private Cloud (Amazon VPC) 資源。若要存取 Apache Kafka 叢集的 VPC，您的來源子網路 EventBridge 需要輸出網際網路存取權。針對公用子網路，這必須是受管理的 NAT 閘道。對於私有子網路，它可以是 NAT 閘道，也可以是您自己的 NAT。確定 NAT 具有公有 IP 地址，可連線至網際網路。

使用下列規則 (最低限度) 設定 Amazon VPC 安全群組：

- 傳入規則 - 針對來源指定的安全群組，允許 Apache Kafka 代理程式連接埠上的所有流量 (9092 代表純文字、9094 代表 TLS、9096 代表 SASL、9098 代表 IAM)。
- 傳出規則：針對所有目的地，允許連接埠 443 上的所有流量。傳入規則：針對來源指定的安全群組，允許 Apache Kafka 代理程式連接埠上的所有流量 (9092 代表純文字、9094 代表 TLS、9096 代表 SASL、9098 代表 IAM)。

使用 Apache 卡夫卡來源的消費者 auto 調整規模

當您最初創建一個 Apache 卡夫卡源，分 EventBridge 配一個消費者來處理卡夫卡主題中的所有分區。每個取用者都有多個並行運行的處理器以處理增加的工作負載。此外，根據工作負載， EventBridge 自動擴展或減少取用者的數量。為了保留每個分割區中的訊息順序，主題中每個分割區的取用者數上限是一個取用者。

在一分鐘間隔內， EventBridge 評估主題中所有分割區的消費者偏移延遲。如果延遲太高，則分區接收消息的速度超 EventBridge 過了處理它們的速度。如有必要，可在主題中 EventBridge 新增或移除取用者。新增或刪除取用者的擴展過程，將在三分鐘的評估期間內完成。

如果您的目標超載， EventBridge 減少消費者的數量。此動作可透過減少取用者可擷取和傳送至函數的訊息數量，減少函數的工作負載。

Amazon Simple Queue Service 當做來源

您可以使用 EventBridge 管道從 Amazon SQS 佇列接收記錄。然後，您可以選擇性地篩選或增強這些記錄，然後再將其傳送至可用的目的地進行處理。

您可以使用管道來處理 Amazon Simple Queue Service (Amazon SQS) 佇列中的訊息。 EventBridge 管道支援標準佇列和先進先出 (FIFO) 佇列。使用 Amazon SQS 時，您可以藉由將任務傳送到佇列並進行非同步處理，以從應用程式中的一個元件中卸載任務。

EventBridge 輪詢佇列，並與包含佇列訊息的事件同步呼叫管道。 EventBridge 批量讀取消息，並為每個批次調用管道一次。當管道成功處理批次時， EventBridge 會從佇列中刪除其訊息。

依預設，會同時 EventBridge 輪詢佇列中最多 10 個訊息，並將該批次傳送至管道。為避免調用具有少量記錄的管道，您可設定批次間隔，要求事件來源緩衝記錄最長達五分鐘。呼叫管道之前，請 EventBridge 繼續輪詢來自 Amazon SQS 標準佇列的訊息，直到發生下列其中一種情況為止：

- 批次化視窗即會到期。
- 已達到調用有效負載大小配額。
- 已達到批次大小可設定的上限。

> **Note**
>
> 如果您使用的是批次視窗，且 Amazon SQS 佇列的流量很低，最多 EventBridge 可能需要等待 20 秒鐘才能叫用管道。即使您將批次時間範圍設定為低於 20 秒也是如此。對於 FIFO 佇列，記錄包含與重複資料刪除和定序相關的其他屬性。

EventBridge 讀取批次時，訊息會保留在佇列中，但會在佇列的可見性逾時長度下隱藏。如果您的管道成功處理批次，請從佇列中 EventBridge 刪除訊息。根據預設，如果管道在處理批次時遇到錯誤，則該批次中的所有訊息會再次顯示在佇列中。因此，您的管道程式碼必須能夠多次處理相同的訊息，而不會產生副作用。您可以在管道回應中包含批次項目失敗的資訊，以便修改此重新處理行為。下列範例顯示兩則訊息之批次的事件。

範例事件

下列範例事件顯示管道接收的資訊。您可以使用此事件來建立和篩選事件模式，或定義輸入轉換。並非所有欄位都可以篩選。如需有關所能篩選欄位的詳細資訊，請參閱 ???。

標準佇列

```
[
  {
    "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
    "receiptHandle": "AQEBwJnKyrHigUMZj6rYigCgxlaS3SLy0a...",
```

```
    "body": "Test message.",
    "attributes": {
      "ApproximateReceiveCount": "1",
      "SentTimestamp": "1545082649183",
      "SenderId": "AIDAIENQZJOLO23YVJ4VO",
      "ApproximateFirstReceiveTimestamp": "1545082649185"
    },
    "messageAttributes": {},
    "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
    "eventSource": "aws:sqs",
    "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:my-queue",
    "awsRegion": "us-east-2"
  },
  {
    "messageId": "2e1424d4-f796-459a-8184-9c92662be6da",
    "receiptHandle": "AQEBzWwaftRI0KuVm4tP+/7q1rGgNqicHq...",
    "body": "Test message.",
    "attributes": {
      "ApproximateReceiveCount": "1",
      "SentTimestamp": "1545082650636",
      "SenderId": "AIDAIENQZJOLO23YVJ4VO",
      "ApproximateFirstReceiveTimestamp": "1545082650649"
    },
    "messageAttributes": {},
    "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
    "eventSource": "aws:sqs",
    "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:my-queue",
    "awsRegion": "us-east-2"
  }
]
```

FIFO 佇列

```
[
  {
    "messageId": "11d6ee51-4cc7-4302-9e22-7cd8afdaadf5",
    "receiptHandle": "AQEBBX8nesZEXmkhsmZeyIE8iQAMig7qw...",
    "body": "Test message.",
    "attributes": {
      "ApproximateReceiveCount": "1",
      "SentTimestamp": "1573251510774",
      "SequenceNumber": "18849496460467696128",
      "MessageGroupId": "1",
```

```
      "SenderId": "AIDAIO23YVJENQZJOL4VO",
      "MessageDeduplicationId": "1",
      "ApproximateFirstReceiveTimestamp": "1573251510774"
    },
    "messageAttributes": {},
    "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
    "eventSource": "aws:sqs",
    "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:fifo.fifo",
    "awsRegion": "us-east-2"
  }
]
```

擴展和處理

對於標準佇列， EventBridge 使用長輪詢來輪詢佇列，直到佇列變為作用中為止。當訊息可用時，最多可 EventBridge 讀取五個批次，並將它們傳送至管道。如果仍然可以使用訊息，則每分鐘最多可 EventBridge 增加 300 個讀取批次的處理序數目。一個管道可同時處理的批次數量上限為 1,000。

對於 FIFO 佇列，請按照接收訊息的順序將訊息 EventBridge 傳送至管道。當您傳送訊息到 FIFO 佇列時，您可以指定訊息群組 ID。Amazon SQS 有助於依序將相同群組中的訊息交付給 EventBridge。EventBridge 將接收的郵件排序為群組，一次只傳送一個群組的批次。如果您的管道傳回錯誤，管道會先嘗試所有受影響訊息的重試，然後再 EventBridge 接收來自同一群組的其他訊息。

配置要與 EventBridge 管道搭配使用的佇列

建立 Amazon SQS 佇列來做為您管道的來源。然後設定佇列，讓管道有時間處理每批事件，並在擴展時重試節流錯誤。 EventBridge

為讓您的管道有時間處理每個記錄批次，請將來源佇列的可見性逾時設定為管道擴充程序和目標元件合併執行期的至少六倍。如果您的管道在處理上一個批次時被節流，則額外的時間允許重試。EventBridge

如果函數多次處理訊息失敗，Amazon SQS 可將訊息傳送到無效字母佇列。當您的管道返回錯誤時，請將其 EventBridge 保留在隊列中。發生可見性逾時之後， EventBridge 會再次收到訊息。若要在多次接收後將訊息傳送至第二個佇列，請在來源佇列上設定無效字母佇列。

> **Note**
>
> 請確定在來源佇列上設定無效字母佇列，而不是在管道上。您在管道上設定的無效字母佇列用於佇列的非同步調用佇列，而不是事件來源佇列。

如果您的管道因為以達到並行上限而傳回錯誤或無法調用，可能可以透過進行額外的嘗試來使處理成功。若要讓訊息在傳送到無效字母佇列前更有機會受到處理，請將來源佇列再驅動政策上的 `maxReceiveCount` 設置值至少為 5。

報告批次項目失敗

當使 EventBridge 用和處理來源的串流資料時，依預設，它會檢查點為批次的最高序號，但只有在批次完全成功時才會檢查點。若要避免重新處理失敗批次中成功處理過的訊息，您可以設定擴充或目標，傳回哪些訊息成功，哪些失敗的物件。我們將其稱為部分批次回應。

如需詳細資訊，請參閱 ???。

成功與失敗條件

如果您傳回下列任一項目，請 EventBridge 將批次視為完全成功：

- 空白 `batchItemFailure` 清單
- Null `batchItemFailure` 清單
- 空白 `EventResponse`
- Null `EventResponse`

如果您傳回下列任一項目，則 EventBridge 會將批次視為完全失敗：

- 空白字串 `itemIdentifier`
- Null `itemIdentifier`
- 具有錯誤金鑰名稱的 `itemIdentifier`

EventBridge 根據您的重試策略重試失敗。

Amazon EventBridge 管道過濾

使用 P EventBridge ipes，您可以篩選指定來源的事件，並僅處理其中的一個子集。此篩選的運作方式與在 EventBridge 事件匯流排或 Lambda 事件來源對應上篩選相同，方法是使用事件模式。如需有關事件模式的詳細資訊，請參閱 ???。

篩選條件標準 `FilterCriteria` 物件是由篩選條件清單 (`Filters`) 組成的結構。每個篩選條件均是定義事件篩選模式 (`Pattern`) 的結構。`Pattern` 是 JSON 篩選條件規則的字串表示法。`FilterCriteria` 物件看起來正如下列範例：

```
{
  "Filters": [
    {"Pattern": "{ \"Metadata1\": [ rule1 ], \"data\": { \"Data1\": [ rule2 ] }}"
    }
  ]
}
```

補充說明，此處是篩選條件的 Pattern 在純文字 JSON 中擴展的值：

```
{
  "Metadata1": [ pattern1 ],
  "data": {"Data1": [ pattern2 ]}
}
```

FilterCriteria 物件共有兩個主要部份：中繼資料屬性和資料屬性。

- 中繼資料屬性是事件物件的欄位。在範例中，FilterCriteria.Metadata1 表示中繼資料屬性。
- 資料屬性是事件主體的欄位。在範例中，FilterCriteria.Data1 表示資料屬性。

例如，假設您的 Kinesis 串流包含如下的事件：

```
{
  "kinesisSchemaVersion": "1.0",
  "partitionKey": "1",
  "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
  "data": {"City": "Seattle",
    "State": "WA",
    "Temperature": "46",
    "Month": "December"
  },
  "approximateArrivalTimestamp": 1545084650.987
}
```

當事件流經您的管道時，使用 base64 編碼的 data 欄位看起來會如下所示：

```
{
  "kinesisSchemaVersion": "1.0",
  "partitionKey": "1",
  "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
  "data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
```

```
  "approximateArrivalTimestamp": 1545084650.987
  "eventSource": "aws:kinesis",
  "eventVersion": "1.0",
  "eventID":
 "shardId-000000000006:49590338271490256608559692538361571095921575989136588898",
  "eventName": "aws:kinesis:record",
  "invokeIdentityArn": "arn:aws:iam::123456789012:role/lambda-role",
  "awsRegion": "us-east-2",
  "eventSourceARN": "arn:aws:kinesis:us-east-2:123456789012:stream/lambda-stream"
},
```

Kinesis 事件上的中繼資料屬性是 `data` 物件外部的任何欄位，例如 `partitionKey` 或 `sequenceNumber`。

Kinesis 事件上的資料屬性是 `data` 物件外部的任何欄位，例如 `City` 或 `Temperature`。

當您進行篩選以符合此事件時，您可以在解碼欄位上使用篩選器。例如，要過濾 `partitionKey` 和 `City`, 您可以使用以下過濾器：

```
{ "partitionKey": [ "1" ], "data": { "City": [ "Seattle" ] }
```

當您建立事件篩選器時，P EventBridge ipes 可以存取事件內容。此內容可能是 JSON 逸出，例如 Amazon SQS body 欄位或以 base64 編碼 (例如 Kinesis `data` 欄位)。如果您的資料是有效的 JSON，您的輸入範本或目標參數的 JSON 路徑可以直接參考內容。例如，如果 Kinesis 事件來源是有效的 JSON，您可以使用 <$.data.someKey> 參考變數。

建立事件模式時，您可以根據來源 API 傳送的欄位進行篩選，而不是輪詢作業新增的欄位。下列欄位不能用於事件模式：

- `awsRegion`
- `eventSource`
- `eventSourceARN`
- `eventVersion`
- `eventID`
- `eventName`
- `invokeIdentityArn`
- `eventSourceKey`

訊息和資料欄位

每個 EventBridge 管道源包含一個包含核心消息或數據的字段。我們將這些稱為消息字段或資料字段。這些字段是特殊的，因為它們可能是 JSON 轉義或 base64 編碼，但是當它們是有效的 JSON 時，它們可以使用 JSON 模式進行過濾，就好像主體未被轉義一樣。這些字段的內容也可以無縫地在輸入變壓器中使用。

正確篩選 Amazon SQS 訊息

如果 Amazon SQS 訊息不符合您的篩選準則， EventBridge 會自動從佇列中移除訊息。您不需要在 Amazon SQS 中手動刪除這些訊息。

針對 Amazon SQS，訊息 body 可以是任何字串。但是，如果您的 `FilterCriteria` 預期 body 為有效的 JSON 格式，這就會造成問題。反之亦然 — 如果傳入的訊息 body 是有效的 JSON 格式，但您的選條件標準預期 body 應為純字串，則此可能會導致意外的行為。

為了避免此問題，請確定 `FilterCriteria` 中的 body 之格式與從佇列收到的訊息中的 body 之預期格式相符。篩選郵件之前， EventBridge 會自動評估內送郵件的格式body和篩選器模式body。如果有不相符項目，請 EventBridge 捨棄訊息。下表摘要說明此評估：

傳入訊息 **body** 格式	篩選條件模式 **body** 格式	產生的動作
純文字的字串	純文字的字串	EventBridge 根據您的篩選條件進行篩選。
純文字的字串	資料屬性沒有篩選條件模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
純文字的字串	有效的 JSON	EventBridge 刪除消息。
有效的 JSON	純文字的字串	EventBridge 刪除消息。
有效的 JSON	資料屬性沒有篩選條件模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
有效的 JSON	有效的 JSON	EventBridge 根據您的篩選條件進行篩選。

如果您不包含body為您的一部分`FilterCriteria`，則 EventBridge 略過此檢查。

正確篩選 Kinesis 和 DynamoDB 訊息

您的篩選條件標準處理 Kinesis 或 DynamoDB 記錄後，串流迭代器將向前移動超過此記錄。如果記錄不滿足篩選條件標準，則無需從事件來源中手動刪除記錄。保留期之後，Kinesis 和 DynamoDB 會自動刪除這些舊記錄。如果希望更快地刪除記錄，請參閱變更資料保留期間。

若要正確篩選來自串流事件來源的事件，資料欄位和資料欄位的篩選條件標準都必須是有效的 JSON 格式。(若為 Kinesis，資料欄位為 `data`。若為 DynamoDB，資料欄位為 `dynamodb`。) 如果其中一個欄位不是有效的 JSON 格式，請 EventBridge 捨棄訊息或擲回例外狀況。下表摘要說明特定行為：

傳入資料格式 (**`data`** 或 **`dynamodb`**)	資料屬性的篩選條件模式格式	產生的動作
有效的 JSON	有效的 JSON	EventBridge 根據您的篩選條件進行篩選。
有效的 JSON	資料屬性沒有篩選條件模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
有效的 JSON	非 JSON	EventBridge 在管道或更新時拋出異常。資料屬性的篩選條件模式必須是有效的 JSON 格式。
非 JSON	有效的 JSON	EventBridge 刪除記錄。
非 JSON	資料屬性沒有篩選條件模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
非 JSON	非 JSON	EventBridge 在建立或更新管道時擲回例外狀況。資料屬性的篩選條件模式必須是有效的 JSON 格式。

針對 Apache Kafka、自我管理的 Apache Kafka 和 Amazon MQ 訊息，正確地篩選由 Amazon 管理的串流

若為 Amazon MQ 來源，訊息欄位為 `data`。若為 Apache Kafka 來源 (Amazon MSK 和自我管理的 Apache Kafka)，則有兩個訊息欄位：`key` 和 `value`。

EventBridge 刪除與過濾器中包含的所有字段不匹配的郵件。對於 Apache Kafka，在成功調用函數後 EventBridge 提交匹配和不匹配的消息的偏移量。對於 Amazon MQ，在成功叫用函數後 EventBridge 確認符合的訊息，並在篩選不相符的訊息時確認這些訊息。

Apache Kafka 和 Amazon MQ 訊息必須是 UTF-8 編碼的字符串，可以是純字串或 JSON 格式。這是因為在套用篩選條件之前，將 Apache 卡夫卡和 Amazon MQ 位元組陣列 EventBridge 解碼為 UTF-8。如果您的郵件使用其他編碼 (例如 UTF-16 或 ASCII)，或郵件格式與格式不相符，則只會 EventBridge 處理中繼資料篩選器。`FilterCriteria`下表摘要說明特定行為：

傳入訊息格式 (**`data`** 或 **`key`** 和 **`value`**)	訊息屬性的篩選條件模式格式	產生的動作
純文字的字串	純文字的字串	EventBridge 根據您的篩選條件進行篩選。
純文字的字串	資料屬性沒有篩選條件模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
純文字的字串	有效的 JSON	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
有效的 JSON	純文字的字串	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。
有效的 JSON	資料屬性沒有篩選條件模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。

傳入訊息格式 (**data** 或 **key** 和 **value**)	訊息屬性的篩選條件模式格式	產生的動作
有效的 JSON	有效的 JSON	EventBridge 根據您的篩選條件進行篩選。
非 UTF-8 編碼字串	JSON、純字串或沒有模式	EventBridge 根據您的篩選條件篩選 (僅限其他中繼資料屬性)。

Lambda ESM 和 EventBridge 管道之間的差異

篩選事件時，Lambda ESM 和 EventBridge 管道的運作方式通常相同。主要區別在於 ESM 有效載荷中不存在該 `eventSourceKey` 欄位。

Amazon EventBridge 管道事件擴充

透過 EventBridge 管道的擴充步驟，您可以在將來源資料傳送到目標之前先增強來源的資料。例如，您可能會收到不包含完整工單資料的票證建立的事件。使用擴充，您可以有一個 Lambda 函數呼叫 `get-ticket` API 以獲取完整的工單詳細信息。然後管道可以將該資訊傳送至目標。

在 EventBridge 中設定管道時，您可以設定下列擴充功能：

- API 目標
- Amazon API Gateway
- Lambda 函數
- Step Functions 狀態機器

> **Note**
>
> EventBridge 管道僅支援快速工作流程做為擴充功能。

EventBridge 會同步調用擴充，因為它必須等待來自擴充的回應，才能調用目標。

擴充回應限制大小為 6MB 以下。

您也可以先轉換從來源接收到的資料，然後再傳送資料以進行增強。如需更多詳細資訊，請參閱 ???。

使用擴充篩選事件

EventBridge 管道會將擴充回應直接傳遞至設定的目標。這包括支援批次的目標的陣列回應。如需批次行為的詳細資訊，請參閱 ???。您也可以使用您的擴充作為篩選器，並傳遞比從來源接收到的事件少。如果您不想調用目標，請返回空響應，例如 ""、{}、或 []。

> **Note**
>
> 如果要使用空有效負載調用目標，請返回具有空 JSON 的數組[{}]。

調用擴充

EventBridge 會同步調用擴充 (調用類型設為 REQUEST_RESPONSE)，因為它必須等待來自擴充的回應，才能調用目標。

> **Note**
>
> 對於 Step Functions 式狀態機器，EventBridge 僅支援 快速工作流程做為擴充功能，因為它們可以同步調用。

Amazon EventBridge 管道目標

您可以將管道中的資料傳送至特定目標。在 EventBridge 中設定管道時，您可以設定下列目標：

- API 目標
- API 閘道
- 批次任務佇列
- CloudWatch 日誌群組
- ECS 任務
- 相同帳戶和地區中的事件匯流排
- Firehose 交付串流
- Inspector 評估範本
- Kinesis 串流

- Lambda 函數 (同步或非同步)
- Redshift 叢集資料 API 查詢
- SageMaker 管道
- Amazon SNS 主題 (不支援 Amazon SNS FIFO 主題)
- Amazon SQS 佇列
- Step Functions 狀態機器
  - 快速工作流程 (同步或非同步)
  - 標準工作流程 (非同步)

目標參數

有些目標服務不會將事件裝載傳送至目標，而是將事件視為叫用特定 API 的觸發器。EventBridge 樑會使用指定 `PipeTargetParameters` 要傳送到該 API 的資訊。所需資訊包括下列項目：

- API 目的地 (傳送至 API 目的地的資料必須與 API 的結構相符。您必須使用 `InputTemplate` 物件來確保資料結構正確。如果您想要包含原始事件裝載，請在 `InputTemplate` 中進行參考。)
- API 閘道 (傳送至 API 閘道的資料必須與 API 的結構相符。您必須使用 `InputTemplate` 物件來確保資料結構正確。如果您想要包含原始事件裝載，請在中 `InputTemplate` 中進行參考。)
- `PipeTargetRedshiftDataParameters` (Amazon Redshift 資料 API 叢集)
- `PipeTargetSageMakerPipelineParameters` (Amazon SageMaker 執行期模型建置管道)
- `PipeTargetBatchJobParameters` (AWS Batch)

> **Note**
>
> EventBridge 不支援所有 JSON 路徑語法，並在執行期對其進行評估。支援的語法包括：
>
> - 點符號 (例如 `$.detail`)
> - 破折號
> - 底線
> - 英數字元
> - 陣列索引
> - 萬用字元 (*)

動態路徑參數

EventBridge 管道目標參數支援選用的動態 JSON 路徑語法。此語法可讓您指定 JSON 路徑，而非靜態值 (例如 `$.detail.state`)。整個值必須是 JSON 路徑，而不僅僅是其中的一部分。例如，`RedshiftParameters.Sql` 可以是 `$.detail.state` 但不能是 `"SELECT * FROM $.detail.state"`。這些路徑會在執行期以指定路徑中的事件裝載本身的資料動態取代。動態路徑參數無法參考輸入轉換所產生的新值或轉換值。動態參數 JSON 路徑支援的語法與轉換輸入時相同。如需更多詳細資訊，請參閱 ???。

動態語法可用於所有 EventBridge 管道擴充的所有字串、非列舉欄位和目標參數，但下列項目除外：

- `PipeTargetCloudWatchLogsParameters.LogStreamName`
- `PipeTargetEventBridgeEventBusParameters.EndpointId`
- `PipeEnrichmentHttpParameters.HeaderParameters`
- `PipeTargetHttpParameters.HeaderParameters`

例如，若要將管道 Kinesis 目標的 `PartitionKey` 設定為來源事件中的自訂金鑰，請將 KinesisTargetParameter.PartitionKey 設定為：

- `"$.data.someKey"` 針對 Kinesis 來源
- `"$.body.someKey"` 針對 Amazon SQS 來源

然後，如果事件掛載為有效 JSON 字串，例如 `{"someKey":"someValue"}`，EventBridge 會從 JSON 路徑擷取該值，並將其用作目標參數。在此範例中，EventBridge 會將 Kinesis `PartitionKey` 設定為`###`。

許可

為了能夠根據您所擁有的資源進行 API 呼叫，EventBridge 管道需要適當的許可。EventBridge 管道會使用您在管道上指定的 IAM 角色，以便使用 IAM 主體 `pipes.amazonaws.com` 進行擴充和鎖定目標呼叫。

調用目標

EventBridge 有下列方法來調用目標：

- 同步處理 (調用類型設定為 `REQUEST_RESPONSE`)：EventBridge 會先等待來自目標的回應，然後再繼續。

- 以非同步方式 (調用類型設定為 FIRE_AND_FORGET)：EventBridge 在繼續之前不會等待回應。

依預設，對於具有排序來源的管道，EventBridge 會同步調用目標，因為在繼續下一個事件之前需要來自目標的回應。

如果來源未強制執行訂單 (例如標準 Amazon SQS 佇列)，EventBridge 可以同步或非同步調用支援的目標。

使用 Lambda 函數和 Step Functions 狀態機器，您可以設定調用類型。

> **Note**
>
> 對於 Step Functions 狀態機器，必須以非同步方式調用標準工作流程。

EventBridge 管道目標細節

AWS Batch 任務佇列

所有 AWS Batch submitJob 參數都使用明確配置 BatchParameters，並且與所有管道 參數一樣，這些參數可以使用傳入事件有效負載的 JSON 路徑進行動態配置。

CloudWatch 日誌群組

無論您是否使用輸入轉換器，事件裝載都會用作日誌訊息。您可以在 PipeTarget 中通過 CloudWatchLogsParameters 設定 Timestamp (或明確 LogStreamName 的目的地)。對於所有管道參數，這些參數可以使用傳入事件有效負載的 JSON 路徑進行動態配置。

Amazon ECS 任務

所有 Amazon ECS runTask 參數都是透過 EcsParameters 明確設定的。對於所有管道參數，這些參數可以使用傳入事件有效負載的 JSON 路徑進行動態配置。

Lambda 函數和 Step Functions 工作流

Lambda 和 Step Functions 沒有批次 API。若要處理來自管道來源的批次事件，批次會轉換為 JSON 陣列，並以輸入的形式傳遞至 Lambda 或 Step Functions 數目標。如需更多詳細資訊，請參閱 ???。

Amazon EventBridge 管道批處理和並發

批次處理行為

EventBridge 管道支援從來源和支援它的目標進行批次處理。此外，還支援 AWS Lambda 和 AWS Step Functions 的批次處理以實現擴充。由於不同的服務支援不同層級的批次處理，因此您無法設定大於目標所支援的批次大小的管道。例如，Amazon Kinesis 串流來源支援的批次大小上限為 10,000 筆記錄，但 Amazon 簡單佇列服務每個批次最多支援 10 則訊息做為目標。因此，從 Kinesis 串流到 Amazon SQS 佇列的管道在來源上可以設定最大批次大小為 10。

如果您使用不支援批次處理的擴充功能或目標來設定管道，您將無法在來源上啟動批次處理。

在來源上啟動批次處理時，JSON 記錄的陣列會透過管道傳遞，然後對應至受支援的擴充或目標的批次 API。輸入轉換器會分別套用於陣列中的每個 JSON 記錄，而不是整個陣列。如需這些陣列的範例，請參閱 ??? 並選取特定來源。即使批次大小為 1，管道也會將批次 API 用於支援的擴充或目標。如果擴充或目標沒有批次 API，但收到完整的 JSON 承載 (例如 Lambda 和 Step 函數)，則會在一個要求中傳送整個 JSON 陣列。即使批次大小為 1，請求也會以 JSON 陣列的形式傳送。

如果在來源設定了管道以進行批次處理，且目標支援批次處理，您可以從擴充中傳回 JSON 項目陣列。此陣列可以包含比原始來源更短或更長的陣列。但是，如果陣列大於目標支持的批次大小，則管道將不會調用目標。

受支援的可分批目標

目標	最大批次大小
CloudWatch 日誌	10,000
EventBridge 事件巴士	10
Firehose 溪	500
Kinesis 串流	500
Lambda 函數	客戶定義
Step Functions 狀態機器	客戶定義
Amazon SNS 主題	10

目標	最大批次大小
Amazon SQS 佇列	10

下列擴充項目和目標會接收完整批次事件承載以進行處理，並受到事件總裝載大小的限制，而不是批次的大小：

- Step Functions 狀態機器 (262144 個字元)
- Lambda 函數 ARN

部分批次失敗

對於 Amazon SQS 和串流來源 (例如 Kinesis 和 DynamoDB)， EventBridge 管道支援目標故障的部分批次失敗處理。如果目標支援批次處理，且只有部分批次成功，則 EventBridge 會自動重試批次處理承載的剩餘部分。對於最 up-to-date 豐富的內容，此重試會透過整個管道進行，包括重新叫用任何已設定的擴充。

不支援擴充的部分批次失敗處理。

對於 Lambda 和 Step Functions 目標，您也可以從目標傳回已定義結構的承載，以指定部分失敗。這表示需要重試的事件。

部分故障有效負載結構示例

```
{
  "batchItemFailures": [
    {
      "itemIdentifier": "id2"
    },
    {
      "itemIdentifier": "id4"
    }
]
```

在此範例中，`itemIdentifier` 會比對目標從其原始來源處理的事件 ID。對於 Amazon SQS 來說，這是 `messageId`. 對於 Kinesis 和 DynamoDB 而言，這是 `eventID`. 若要讓 P EventBridge ipes 充分處理來自目標的部分批次失敗，這些欄位必須包含在擴充傳回的任何陣列裝載中。

輸送量和並發行為

管道接收到的每個事件或一批事件，而該事件傳遞到一個擴充或目標都被視為管道執行。處於 STARTED 狀態的管道會持續輪詢來自來源的事件，並根據可用的積壓和配置的批次處理設定擴展和縮減。

有關並行管道執行的配額，以及每個帳戶和區域的管道數量，請參閱 ???。

根據預設，根據來源，單一管道會擴充至下列最大並行執行次數：

- DynamoDB：並行執行可以攀升至管道上 ParallelizationFactor 設定的高度乘以串流中的碎片數目。
- Apache Kafka：並行執行可以攀升與該主題有關的分區數量一樣高，最多可達 1000 個。
- Kinesis：並行執行可以攀升至管道上 ParallelizationFactor 設定的高度乘以串流中的碎片數目。
- Amazon MQ：5
- Amazon SQS：1250

如果您需要更高的最大輪詢輸送量或並行限制，請聯絡支援人員。

> **Note**
>
> 執行限制被認為是盡力而為的安全限制。雖然輪詢不會限制在這些值之下，但管道或帳戶可能會高於這些建議值。

管道執行的時間限制為最多 5 分鐘，包括擴充和目標處理。此限制目前無法提高。

具有嚴格排序來源 (例如 Amazon SQS FIFO 佇列、Kinesis 和 DynamoDB Streams 或 Apache Kafka 主題) 的管道會進一步受到來源組態的並行限制，例如 FIFO 佇列的訊息群組 ID 數目或 Kinesis 佇列的碎片數目。由於在這些限制範圍內嚴格保證訂購，因此具有有序源的管道不能超過這些並發限制。

Amazon EventBridge 管道輸入轉換

Amazon EventBridge 管道在將資料傳遞至擴充功能和目標時，支援選用的輸入轉換器。您可以使用輸入轉換器來重塑 JSON 事件輸入裝載，以滿足擴充或目標服務的需求。對於 Amazon API Gateway 和 API 目的地，這是您將輸入事件塑造為 API 的 RESTful 模型的方式。輸入轉換器被建模為 InputTemplate 參數。它們可以是自由文字、事件承載的 JSON 路徑，或包含事件承載之內嵌

JSON 路徑的 JSON 物件。對於擴充，事件裝載來自來源。對於目標而言，事件有效負載是從擴充傳回的 (如果在管道上設定)。除了事件承載中的服務特定資料之外，您還可以在您的 `InputTemplate` 中使用保留變數來參考管道的資料。

若要存取陣列中的項目，請使用方括號標記法。

> **Note**
>
> EventBridge 不支援所有 JSON 路徑語法，並在執行期對其進行評估。支援的語法包括：
>
> - 點符號 (例如 `$.detail`)
> - 破折號
> - 底線
> - 英數字元
> - 陣列索引
> - 萬用字元 (*)

以下是參考 Amazon SQS 事件承載的範例 `InputTemplate` 參數：

靜態字符串

```
InputTemplate: "Hello, sender"
```

JSON 路徑

```
InputTemplate: <$.attributes.SenderId>
```

動態字符串

```
InputTemplate: "Hello, <$.attributes.SenderId>"
```

靜態 JSON

```
InputTemplate: >
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3",
```

```
}
```

動態 JSON

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.key>,
  "d": <aws.pipes.event.ingestion-time>
}
```

使用方括號標記法存取陣列中的項目：

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.Records[3]>,
  "d": <aws.pipes.event.ingestion-time>
}
```

> **Note**
>
> EventBridge 在執行期替換輸入轉換器，以確保有效的 JSON 輸出。因此，請在參考 JSON 路徑參數的變數周圍加上引號，但不要在參考 JSON 物件或陣列的變數周圍加上引號。

預留變數

輸入範本可以使用以下預留變數：

- `<aws.pipes.pipe-arn>`：管道的 Amazon Resource Name (ARN)。
- `<aws.pipes.pipe-name>`：管道的名稱。
- `<aws.pipes.source-arn>`：管道之事件來源的 ARN。
- `<aws.pipes.enrichment-arn>`：管道的擴充 ARN。
- `<aws.pipes.target-arn>`：管道目標的 ARN。
- `<aws.pipes.event.ingestion-time>`：輸入變壓器所收到事件的時間。這是一個 ISO 8601 時間戳記。此時間對於擴充輸入變壓器和目標輸入變壓器而有所不同，具體取決於擴充完成處理事件的時間。

- `<aws.pipes.event>`：輸入變壓器所收到的事件。

  對於擴充輸入變壓器，這是來源的事件。這包含來源的原始裝載，加上額外的服務特定中繼資料。如需服務的特定範例，請參閱 ??? 中的主題。

  對於目標輸入轉換器，這是擴充所傳回的事件 (如果已設定)，不含其他中繼資料。因此，擴充傳回的承載可能是非 JSON。如果管道上未設定擴充，則這是來自具有中繼資料的來源的事件。
- `<aws.pipes.event.json>`：與 `aws.pipes.event` 相同，但如果原始有效負載 (來源或由擴充傳回) 為 JSON，則變數只有值。如果管道具有編碼欄位 (例如 Amazon SQS body 欄位或 Kinesis) `data`，則會將這些欄位解碼並轉換為有效的 JSON。因為它不會逸出，因此變數只能用作 JSON 欄位的值。如需更多詳細資訊，請參閱 ???。

輸入轉換範例

以下是我們可以用作範例事件的 Amazon EC2 事件範例。

```
{
  "version": "0",
  "id": "7bf73129-1428-4cd3-a780-95db273d1602",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "123456789012",
  "time": "2015-11-11T21:29:54Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"
  ],
  "detail": {
    "instance-id": "i-0123456789",
    "state": "RUNNING"
  }
}
```

讓我們使用下面的 JSON 作為我們的轉換器。

```
{
  "instance" : <$.detail.instance-id>,
  "state": <$.detail.state>,
  "pipeArn" : <aws.pipes.pipe-arn>,
```

```
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
}
```

以下為其輸出結果：

```
{
  "instance" : "i-0123456789",
  "state": "RUNNING",
  "pipeArn" : "arn:aws:pipe:us-east-1:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

隱式主體數據解析

傳入裝載中的下列欄位可能是 JSON 逸出，例如 Amazon SQS body 物件，或是以 base64 編碼的欄位 (例如 Kinesis `data` 物件)。對於篩選和輸入轉換，EventBridge 會將這些欄位轉換為有效的 JSON，以便直接參考子值。例如，`<$.data.someKey>` 針對 Kinesis。

若要讓目標在沒有任何其他中繼資料的情況下接收原始有效負載，請使用輸入轉換器與此主體資料 (特定於來源)。例如，`<$.body>` 對於 Amazon SQS，或 `<$.data>` 對於Kinesis。如果原始裝載是有效的 JSON 字串 (例如 `{"key": "value"}`)，則搭配來源特定主體資料使用輸入轉換器會導致原始來源裝載中的引號遭到移除。例如，`{"key": "value"}` 傳送至目標時會變成 `"{key: value}"`。如果您的目標需要有效的 JSON 承載資料 (例如，EventBridge Lambda 或 Step Functions)，這將導致傳遞失敗。若要讓目標接收原始來源資料而不產生無效的 JSON，請將來源主體資料輸入轉換器包裝在 JSON 中。例如：`{"data": <$.data>}`。

隱含主體剖析也可用於動態填入大多數管道目標或擴充參數的值。如需詳細資訊，請參閱 ???

> **Note**
>
> 如果原始承載是有效的 JSON，此欄位將包含未逸出、非 base64 編碼的 JSON。但是，如果承載不是有效的 JSON，則 EventBridge base64 編碼以下列出的欄位，Amazon SQS 除外。

- 作用中的 MQ：`data`
- kinesis：`data`

- Amazon MSK：`key` 和 `value`
- Rabbit MQ：`data`
- 自我管理的 Apache Kafka;：`key` 和 `value`
- Amazon SQS：`body`

轉換輸入的常見問題

這些是在 EventBridge 管道中轉換輸入時的一些常見問題：

- 針對字串，引號是必要的。
- 為您的範本建立 JSON 路徑時沒有驗證。
- 如果您指定變數對應不存在於事件中的 JSON 路徑，則該變數不會建立，且不會出現在輸出中。
- 像 `aws.pipes.event.json` 的JSON 屬性只能用作 JSON 欄位的值，而不能內嵌在其他字串中。
- 當填入目標的輸入範本時，EventBridge 不會逸出由輸入路徑擷取的值。
- 如果 JSON 路徑參考 JSON 物件或陣列，但該變數在字串中參照，EventBridge 會移除任何內部引號，以確保有效的字串。例如，「主體為 <$ .body >」會導致 EventBridge 移除物件中的引號。

  因此，如果您想要根據單一 JSON 路徑變數來輸出 JSON 物件，則必須將其做為索引鍵放置。就本範例而言，`{"body": <$.body>}`。
- 代表字串的變數不需要引號。它們是允許的，但是 EventBridge 管道會在轉換期間自動將引號加入字串變數值，以確保轉換輸出是有效的 JSON。EventBridge 接管道不會在代表 JSON 物件或陣列的變數中加入引號。請勿在代表 JSON 物件或陣列的變數中加入引號。

  例如，下列輸入範本包含代表字串和 JSON 物件的變數：

```
{
  "pipeArn" : <aws.pipes.pipe-arn>,
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
}
```

  使用適當的引號導致有效的 JSON：

```
{
  "pipeArn" : "arn:aws:events:us-east-2:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
```

```
    ... // commented for brevity
  }
}
```

- 對於 Lambda 或 Step Functions 擴充或目標，即使批次大小為 1，批次也會以 JSON 陣列的形式傳送至目標。但是，輸入轉換器仍將應用於 JSON Array 中的單個記錄，而不是整個數組。如需更多詳細資訊，請參閱 ???。

日誌 Amazon EventBridge 管道

EventBridge 管道記錄可讓「管 EventBridge 道」將詳細管道效能的記錄傳送至支援的 AWS 服務。使用日誌來深入瞭解管道的執行效能，並協助進行疑難排解和偵錯。

您可以選取下列 AWS 服務作為 P EventBridge ipes 傳送記錄的記錄目標：

- CloudWatch 日誌

  EventBridge 會將記錄 CloudWatch 檔記錄傳送至指定的記錄檔記錄群組。

  使用 CloudWatch Logs 將您使用的所有系統、應用程式和 AWS 服務的記錄集中在單一、可高度擴充的服務中。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的使用日誌群組和日誌串流。

- Firehose 流原木

  EventBridge 將日誌記錄傳送到 Firehose 交付流。

  Amazon Data Firehose 是一項全受管服務，可將即時串流資料傳送至特定服 AWS 務等目的地，以及支援的第三方服務供應商擁有的任何自訂 HTTP 端點或 HTTP 端點。如需詳細資訊，請參閱 Amazon 資料 Firehose 使用者指南中的建立 Amazon 資料 Firehose 交付串流。

- Amazon S3 日誌

  EventBridge 以 Amazon S3 物件的形式將日誌記錄傳送到指定的儲存貯體。

  Amazon S3 是一項物件儲存服務，提供領先業界的可擴展性、資料可用性、安全性和效能。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 中的上傳、下載和使用 Amazon S3 中的物件。

Amazon EventBridge 管道日誌如何工作

管道接收到的每個事件或一批事件，而該事件傳遞到一個擴充和/或目標都被視為管道執行。如果啟用，則 EventBridge 會在處理事件批次時為其執行的每個執行步驟產生記錄。記錄中包含的資訊會套用至事件批次，無論是單一事件還是最多 10,000 個事件。

您可以在管道來源和目標上設定事件批次的大小。如需詳細資訊，請參閱 ???。

傳送至每個日誌目的地的記錄資料相同。

如果已設定 Amazon CloudWatch 日誌目的地，傳送到所有目的地的日誌記錄的限制為 256kb。欄位將視需要截斷。

您可以使用下列方式自訂 EventBridge 傳送至所選記錄目的地的記錄：

- 您可以指定記錄層級，以決定 EventBridge 將記錄傳送至所選記錄目的地的執行步驟。如需詳細資訊，請參閱 ???。
- 您可以指定 P EventBridge ipes 是否在相關執行步驟的記錄中包含執行資料。此資料包括：
  - 事件批次的裝載
  - 傳送至 AWS 擴充或目標服務的要求
  - AWS 擴充或目標服務傳回的回應

  如需詳細資訊，請參閱 ???。

指定 EventBridge 管道記錄層級

您可以指定 EventBridge 將記錄傳送至所選記錄目的地的執行步驟類型。

請從下列詳細層次中選擇要包含在日誌記錄中的細節層次。日誌層級會套用至為管道指定的所有日誌目的地。每個日誌層級都包含先前日誌層級的執行步驟。

- 關閉 — EventBridge 不傳送任何記錄到任何指定的記錄目的地。這是預設設定。
- ERROR — EventBridge 將與管道執行期間產生的錯誤相關的任何記錄傳送至指定的記錄目的地。
- INFO — EventBridge 傳送與錯誤相關的任何記錄，以及選取管道執行期間執行的其他步驟至指定的記錄目的地。
- TRACE — EventBridge 將在管道執行任何步驟期間產生的任何記錄傳送至指定的記錄目的地。

在 EventBridge 主控台中，預設會選取 CloudWatch 記錄檔做為記錄目的地，與記ERROR錄層級一樣。因此，依預設，P EventBridge ipes 會建立新的記 CloudWatch 錄群組，將包含詳細資料ERROR層級的記錄檔記錄傳送至該群組。以程式設計方式設定日誌記錄檔時，不會選取預設值。

下表列出了每個日誌層級中包含的執行步驟。

步驟	TRACE	INFO	ERROR	OFF
執行已失敗	x	x	x	
部分執行失敗	x	x	x	
執行已開始	x	x		
執行已成功	x	x		
限制執行	x	x	x	
執行逾時	x	x	x	
擴展調用失敗	x	x	x	
已跳過的擴充調用	x	x		
擴充調用已開始	x			
擴充調用已成功	x			
已進入擴充階段	x	x		
擴充階段失敗	x	x	x	
擴充階段成功	x	x		
擴充轉換失敗	x	x	x	
擴充轉換開始	x			
擴充轉換成功	x			
目標調用失敗	x	x	x	

步驟	TRACE	INFO	ERROR	OFF
目標調用部分失敗	x	x	x	
目標調用已跳過	x			
目標調用已開始	x			
目標調用成功	x			
已進入目標階段	x	x		
目標階段失敗	x	x	x	
目標階段部分失敗	x	x	x	
跳過的目標階段	x			
目標階段成功	x	x		
目標轉換失敗	x	x	x	
目標轉換已開始	x			
目標轉換成功	x			

在 EventBridge 管道記錄中包含執行資料

您可以指定為 EventBridge 將執行資料包含在其產生的記錄中。執行資料包括代表事件批次裝載的欄位，以及傳送至的要求以及來自擴充和目標的回應的欄位。

執行資料對於疑難排解和偵錯很有用。該 `payload` 欄位包含批次中包含的每個事件的實際內容，可讓您將個別事件與特定管路執行相關聯。

如果您選擇包括執行資料，則會針對管道指定的所有日誌目的地加入該資料。

> **Important**
>
> 這些欄位可能包含敏感資訊。 EventBridge 不會嘗試在記錄期間編輯這些欄位的內容。

包含執行資料時，會 EventBridge 將下列欄位新增至相關記錄：

- **payload**

  表示管道正在處理的事件批次的內容。

  EventBridge 在可能已更新事件批次內容的步驟所產生的記錄中包含payload欄位。這包括以下步驟：

  - EXECUTION_STARTED
  - ENRICHMENT_TRANSFORMATION_SUCCEEDED
  - ENRICHMENT_STAGE_SUCCEEDED
  - TARGET_TRANSFORMATION_SUCCEEDED
  - TARGET_STAGE_SUCCEEDED

- **awsRequest**

  代表傳送至擴充或目標做為 JSON 字串傳送的請求。對於傳送至 API 目的地的要求，這代表傳送至該端點的 HTTP 要求。

  EventBridge 在擴充和鎖定目標的最後步驟所產生的記錄中包含awsRequest欄位； EventBridge 亦即，在針對指定的擴充或目標服務執行要求之後，或嘗試執行要求之後。這包括以下步驟：

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED
  - TARGET_INVOCATION_PARTIALLY_FAILED
  - TARGET_INVOCATION_SUCCEEDED

- **awsResponse**

  以 JSON 格式表示擴充或目標傳回的回應。對於傳送至 API 目的地的要求，這代表傳送至該端點的 HTTP 要求。

  如同awsRequest，在擴充和鎖定目標的最後步驟所產生的記錄中 EventBridge 包含awsResponse欄位； EventBridge 亦即，在針對指定的擴充或目標服務執行要求之後，或嘗試執行要求之後，並收到回應。這包括以下步驟：

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED

- TARGET_INVOCATION_PARTIALLY_FAILED
- TARGET_INVOCATION_SUCCEEDED

如需管道執行步驟的討論，請參閱 ???。

截斷 EventBridge管道記錄中的執行資料

如果您選擇在管道的記錄檔記錄中 EventBridge 包含執行資料，則記錄可能超過 256 KB 的大小限制。若要避免這種情況發生，請依照下列順序 EventBridge自動截斷執行資料欄位。 EventBridge 在進行截斷下一個欄位之前，會完全截斷每個欄位。 EventBridge 只要移除資料字串結尾的字元，就會截斷欄位資料；不會嘗試根據資料重要性進行截斷，而截斷會使 JSON 格式無效。

- payload
- awsRequest
- awsResponse

如果 EventBridge 不截斷事件中的欄位，則truncatedFields欄位會包含截斷資料欄位的清單。

EventBridge 管道記錄中的錯誤報告

EventBridge 也會在代表失敗狀態的管道執行步驟中包含錯誤資料 (如果有的話)。這些步驟包括：

- ExecutionThrottled
- ExecutionTimeout
- ExecutionFailed
- ExecutionPartiallyFailed
- EnrichmentTransformationFailed
- EnrichmentInvocationFailed
- EnrichmentStageFailed
- TargetTransformationFailed
- TargetInvocationFailed
- TargetInvocationPartiallyFailed
- TargetStageFailed
- TargetStagePartiallyFailed

EventBridge 管道執行步驟

瞭解管道執行步驟的流程可協助您使用日誌進行管道效能的疑難排解或偵錯。

管道接收到的每個事件或一批事件，而該事件傳遞到一個擴充或目標都被視為管道執行。如果啟用，則 EventBridge 會在處理事件批次時為其執行的每個執行步驟產生記錄。

在高層級，執行包含兩個階段或步驟集合：擴充和目標。這些階段中的每一個都包含轉換和調用步驟。

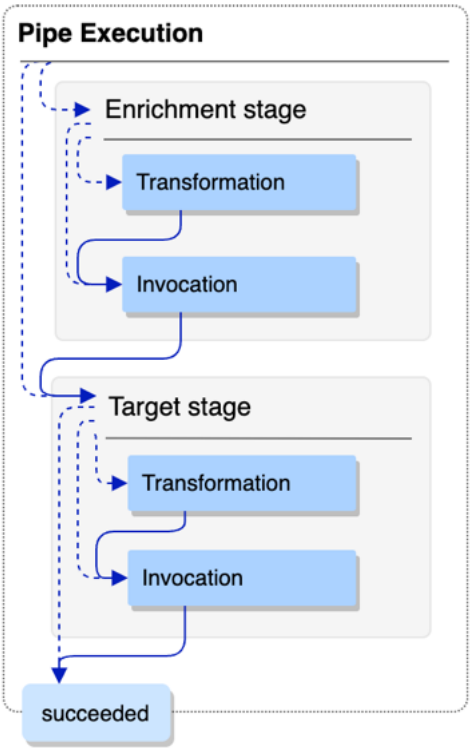
成功執行管道的主要步驟如下：

- 管道執行開始。
- 如果您已經為事件指定了擴充，則執行會進入擴充階段。如果您尚未指定擴充，執行會繼續到目標階段。

  在擴充階段中，管道會執行您指定的任何轉換，然後調用擴充。
- 在目標階段中，管道會執行您指定的任何轉換，然後調用目標。

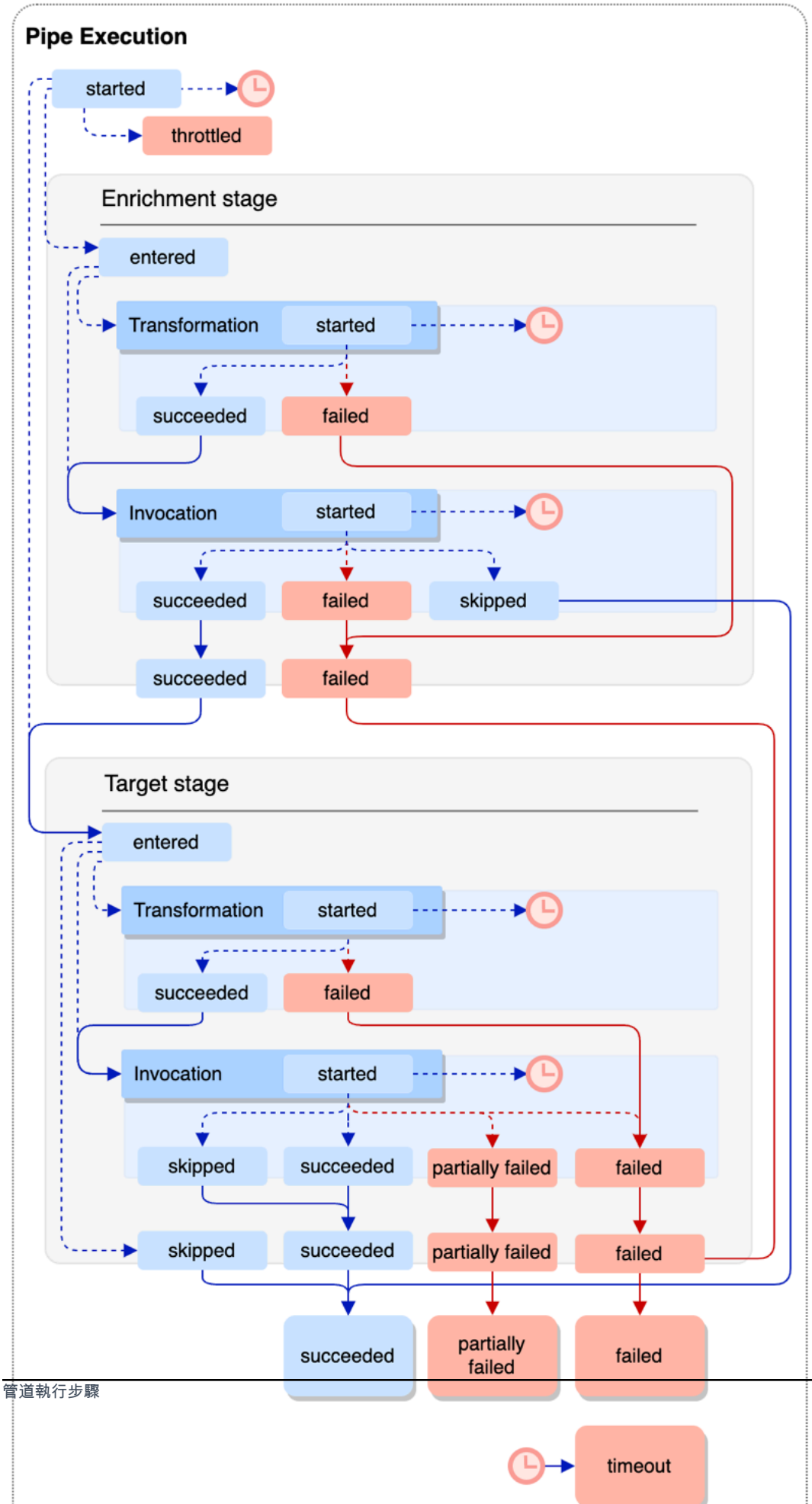
  如果您尚未指定轉換或目標，則執行會略過目標階段。
- 管執行成功完成。

下圖演示了此流程。發散路徑會格式化為虛線。



下圖顯示了管道執行流程的詳細視圖，並表示了所有可能的執行步驟。發散路徑會格式化為虛線。

如需管道執行步驟的完整清單，請參閱 ???。


Pipe Execution
started
throttled
Enrichment stage
entered
Transformation
started
succeeded
failed
Invocation
started
succeeded
failed
skipped
succeeded
failed
Target stage
entered
Transformation
started
succeeded
failed
Invocation
started
skipped
succeeded
partially failed
failed
skipped
succeeded
partially failed
failed
succeeded
partially failed
failed
timeout

請注意，目標調用可能會導致批次發生部分失敗。如需詳細資訊，請參閱 ???。

EventBridge 管道記錄綱要參考

以下參考資料詳細說明了 EventBridge 管道記錄的結構描述。

每個日誌記錄代表一個管道執行步驟，如果管道來源和目標已設定為批次處理，則可能包含多達 10,000 個事件。

如需詳細資訊，請參閱 ???。

```
{
    "executionId": "guid",
    "timestamp": "date_time",
    "messageType": "execution_step",
    "resourceArn": "arn:aws:pipes:region:account:pipe/pipe-name",
    "logLevel": "TRACE | INFO | ERROR",
    "payload": "{}",
    "awsRequest": "{}"
    "awsResponse":"{}"
    "truncatedFields": ["awsRequest","awsResponse","payload"],
    "error": {
        "httpStatusCode": code,
        "message": "error_message",
        "details": "",
        "awsService": "service_name",
        "requestId": "service_request_id"
    }
}
```

executionId

管道執行的識別碼。

管道接收到的每個事件或一批事件，而該事件傳遞到一個擴充或目標都被視為管道執行。如需詳細資訊，請參閱 ???。

timestamp

發出日誌事件的日期和時間。

單位：毫秒。

messageType

為其產生記錄的管道執行步驟。

如需執行步驟的詳細資訊，請參閱 ???。

resourceArn

管道的 Amazon Resource Name (ARN)。

logLevel

為管道日誌指定的詳細資料層級。

有效值：ERROR | INFO | TRACE

如需詳細資訊，請參閱 ???。

payload

管道正在處理的事件批次的內容。

EventBridge 僅當您指定在此管道的記錄中包含執行資料時，才會包含此欄位。如需更多資訊，請參閱 ???

> **Important**
> 這些欄位可能包含敏感資訊。 EventBridge 不會嘗試在記錄期間編輯這些欄位的內容。

如需詳細資訊，請參閱 ???。

awsRequest

以 JSON 格式傳送至擴充或目標服務的要求。對於傳送至 API 目的地的要求，這代表傳送至該端點的 HTTP 要求。

EventBridge 僅當您指定在此管道的記錄中包含執行資料時，才會包含此欄位。如需更多資訊，請參閱 ???

> **Important**
> 這些欄位可能包含敏感資訊。 EventBridge 不會嘗試在記錄期間編輯這些欄位的內容。

如需詳細資訊，請參閱 ???。

awsResponse

以 JSON 格式表示擴充或目標傳回的回應。對於傳送至 API 目的地的要求，這代表從該端點傳回的 HTTP 回應，而不是 API 目的地服務本身傳回的回應。

EventBridge 僅當您指定在此管道的記錄中包含執行資料時，才會包含此欄位。如需更多資訊，請參閱 ???

> **Important**
>
> 這些欄位可能包含敏感資訊。 EventBridge 不會嘗試在記錄期間編輯這些欄位的內容。

如需詳細資訊，請參閱 ???。

truncatedFields

EventBridge 已截斷任何執行資料欄位的清單，以保持在 256 KB 大小限制以下的記錄。

如果 EventBridge 不需要截斷任何執行資料欄位，則此欄位存在，但是。`null`

如需詳細資訊，請參閱 ???。

error

包含在此管道執行步驟期間產生之任何錯誤的資訊。

如果在此管道執行步驟期間未產生錯誤，則此欄位存在，除了 `null`。

httpStatusCode

被呼叫的服務傳回的 HTTP 狀態碼。

message

呼叫的服務傳回的錯誤訊息。

詳細資訊

被呼叫的服務傳回的任何詳細錯誤資訊。

awsService

被呼叫的服務的名稱。

requestId

被呼叫的服務發出此請求的 ID。

使用和 Amazon CloudWatch 日誌記錄 AWS CloudTrail 和監控 Amazon EventBridge 管道

您可以記錄 P EventBridge ipes 呼叫，並使用指標來使用 CloudTrail 和監視管道的健全狀況。CloudWatch

CloudWatch 度量

EventBridge Pipes CloudWatch 每分鐘都會將指標傳送到 Amazon，從限制的管道執行到成功叫用的目標等所有內容。

指標	描述
`Concurrency`	一個管道並行執行的數目。 有效尺寸: AwsAccountId 單位：無
`Duration`	管道執行所花費的時間長度。 有效尺寸: PipeName 單位：毫秒
`EventCount`	管道已處理的事件數目。 有效尺寸: PipeName 單位：無
`EventSize`	調用管道之事件的有效負載大小。 有效尺寸: PipeName 單位：位元組

指標	描述
`ExecutionThrottled`	限流了多少次執行管道。 **Note** 如果沒有限制執行，則此值將為 `0`。 有效尺寸: AwsAccountId, PipeName 單位：無
`ExecutionTimeout`	在完成執行之前，管道的執行有多少次逾時。 **Note** 如果沒有執行逾時，則此值為 `0`。 有效尺寸: PipeName 單位：無
`ExecutionFailed`	有多少管道執行失敗。 **Note** 如果沒有執行失敗，則此值為 `0`。 有效尺寸: PipeName 單位：無

指標	描述
ExecutionPartiallyFailed	多少管道執行部分失敗。 Note 如果沒有部分執行失敗，則此值為 0。 有效尺寸: PipeName 單位：無
EnrichmentStageDuration	擴充階段花了多長時間才能完成。 有效尺寸: PipeName 單位：毫秒
EnrichmentStageFailed	有多少個管道的擴充階段的執行失敗。 Note 如果沒有執行失敗，則此值為 0。 有效尺寸: PipeName 單位：無
Invocations	調用的總數目。 有效尺寸: AwsAccountId, PipeName 單位：無
TargetStageDuration	目標階段需要多長時間才能完成。 有效尺寸: PipeName 單位：毫秒

指標	描述
TargetStageFailed	管道的目標階段執行失敗了多少次。 **Note** 如果沒有執行失敗，則此值為 0。 有效尺寸: PipeName 單位：無
TargetStagePartiallyFailed	有多少管道目標階段的執行部分失敗。 **Note** 如果沒有部分執行失敗，則此值為 0。 有效尺寸: PipeName 單位：無
TargetStageSkipped	有多少管道目標階段的執行被跳過 (例如，由於擴充返回空的有效負載)。 有效尺寸: PipeName 單位：計數

量度的維 CloudWatch 度

CloudWatch 量度具有維度或可排序屬性，如下所示。

維度	描述
AwsAccountId	依用戶身分篩選可用的指標。
PipeName	依管道名稱篩選可用的指標。

Amazon EventBridge 管道錯誤處理和故障排除

重試行為和錯誤處理

針對來源服務、擴充或目標服務或 EventBridge 的任何可重試 AWS 失敗，EventBridge 管道會自動重試擴充和目標調用。但是，如果擴充或目標客戶實作傳回失敗，則管道輪詢輸送量將逐漸退回。對於幾乎連續的 4xx 錯誤 (例如 IAM 的授權問題或缺少資源)，管道可以自動停用，並在 `StateReason` 中帶有解釋訊息。

管道調用錯誤和重試行為

當您調用管道時，可能會發生兩種主要類型的錯誤：管道內部錯誤和客戶調用錯誤。

管內部錯誤

管道內部錯誤是由 EventBridge 管道服務管理的調用方面所造成的錯誤。

這些類型的錯誤可能包括以下問題：

- 嘗試調用客戶 Targer 服務時發生 HTTP 連線失敗
- 管道服務本身的可用性暫時下降。

一般而言，EventBridge 管道會無限次重試內部錯誤，並且只有在來源中的記錄到期時才會停止。

對於具有串流來源的管道，EventBridge Pipes 不會將內部錯誤的重試次數與串流來源的重試政策上指定的最大重試次數計算。對於具有 Amazon SQS 來源的管道，EventBridge 管道不會將內部錯誤的重試次數與 Amazon SQS 來源的最大接收次數計算。

客戶調用錯誤

客戶調用錯誤是由用戶管理的配置或代碼引起的錯誤。

這些類型的錯誤可能包括以下問題：

- 管道的權限不足，無法調用目標。
- 同步調用的客戶 Lambda、Step Functions、API 目標或 API 閘道端點中的邏輯錯誤。

針對客戶調用錯誤，EventBridge 管道會執行下列動作：

- 對於具有串流來源的管道，EventBridge 管道會重試直到管道重試原則上設定的重試次數上限，或直到記錄保留天數上限到期為止 (以先到者為準)。
- 對於具有 Amazon SQS 來源的管道，EventBridge 管道會重試客戶錯誤，直到來源佇列上的最大接收計數為止。
- 對於具有 Apache Kafka 或 Amazon MQ 來源的管道，EventBridge 會重試客戶錯誤，因為它會重試內部錯誤。

對於具有計算目標的管道，您必須同步調用管道，以便 EventBridge 管道知道從客戶計算邏輯擲回的任何執行期錯誤，然後重試此類錯誤。管道無法重試從 Step Functions 標準工作流程邏輯擲回的錯誤，因為必須以非同步方式調用此目標。

對於 Amazon SQS 和串流來源 (例如 Kinesis 和 DynamoDB)，EventBridge 管道支援目標故障的部分批次失敗處理。如需詳細資訊，請參閱部分批次失敗。

管道 DLQ 行為

管道從來源繼承無效字母佇列 (DLQ) 行為：

- 如果來源 Amazon SQS 佇列具有已設定的 DLQ，則訊息會在指定的嘗試次數後自動傳送至該處。
- 對於串流來源 (例如 DynamoDB 和 Kinesis 串流)，您可以為管道和路由事件設定 DLQ。DynamoDB 和 Kinesis 串流來源支援 Amazon SQS 佇列和 Amazon SNS 主題做為 DLQ 目標。

如果您為具有 Kinesis 或 DynamoDB 來源的管路指定 `DeadLetterConfig`，請確定管道上的 `MaximumRecordAgeInSeconds` 屬性小於來源事件的 `MaximumRecordAge` 屬性。`MaximumRecordAgeInSeconds` 控制管道輪詢器何時放棄事件並將其傳遞給 DLQ，並且 `MaximumRecordAge` 控制在源流中可見的消息多長時間才被刪除。因此，請設定 `MaximumRecordAgeInSeconds` 為小於 `MaximumRecordAge` 來源的值，以便在事件傳送至 DLQ 之間有足夠的時間，以及來源自動刪除，以便您判斷事件為何進入 DLQ。

對於 Amazon MQ 來源，DLQ 可以直接在訊息代理程式上設定。

EventBridge 管道不支援串流來源的先進先出 (FIFO) DLQ。

EventBridge 管道不支援 Amazon MSK 串流和自我管理的 Apache Kafka 串流來源的 DLQ。

管路故障狀態

建立、刪除和更新管道是非同步作業，可能會導致失敗狀態。同樣地，管道可能會因故障而自動停用。在所有情況下，管道 `StateReason` 都會提供協助疑難排解故障的資訊。

以下是 StateReason 可能值的清單範例：

- 找不到串流。若要繼續處理，請刪除並建立新的管道。
- 管道沒有執行佇列作業所需的權限 (sqs:ReceiveMessage, sqs:DeleteMessage and sqs:GetQueueAttributes)
- 連線錯誤。您的 VPC 必須能夠連線至管道。您可以透過設定 NAT 閘道來提供存取權。有關如何設置 NAT 閘道，請查看 AWS 文檔。
- MSK 叢集沒有與其相關聯的安全群組

管道可能會隨著更新而自動停止 StateReason。可能的原因包括：

- 設定為擴充的 Step Functions 標準工作流程。
- 設定為要同步調用之目標的「步驟函數」標準工作流程。

自訂加密失敗

如果將來源設定為使用 AWS KMS 自訂加密金鑰 (CMK)，而不是 AWS 管控道 AWS KMS 金鑰，則必須明確授與管道的執行角色解密權限。若要這麼做，請在自訂 CMK 原則中包含下列額外權限：

```
  {
      "Sid": "Allow Pipes access",
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::01234567890:role/service-role/
Amazon_EventBridge_Pipe_DDBStreamSourcePipe_12345678"
      },
      "Action": "kms:Decrypt",
      "Resource": "*"
  }
```

將上述角色替換為管道的執行角色。

所有帶有 AWS KMS CMK 的管道來源都是如此，包括：

- Amazon DynamoDB Streams
- Amazon Kinesis Data Streams
- Amazon MQ

- Amazon MSK
- Amazon SQS

教學課程：建立可篩選來源事件的 EventBridge 管道

在本教學中，您將建立一個管道，將 DynamoDB 串流來源連接到 Amazon SQS 佇列目標。這包括為管道指定事件模式，以便在篩選要傳遞至佇列的事件時使用。然後，您將測試管道，以確保只傳遞所需的事件。

先決條件：建立來源和目標

在建立管之前，您需要建立管要連接的來源和目標。在此情況下，Amazon DynamoDB 資料串流可做為管道來源，以及 Amazon SQS 佇列做為管道目標。

若要簡化此步驟，您可以使用 AWS CloudFormation 佈建來源和目標資源。若要這麼做，您將建立定義下列資源的 CloudFormation 範本：

- 管道源

  名為 `pipe-tutorial-source` 的 Amazon DynamoDB 資料表使用啟用串流，以提供 DynamoDB 資料表中項目變更資訊的排序流程。

- 管道類型

  一個名為 `pipe-tutorial-target` 的 Amazon SQS 佇列，用於從您的管道接收 DynamoDB 事件串流。

若要建立用於佈建管道資源的 CloudFormation 範本

1. 複製下方 ??? 區段中的 JSON 範本文字。
2. 將範本儲存為 JSON 檔案 (例如 `~/pipe-tutorial-resources.json`)。

接下來，使用您剛建立的範本檔案來佈建 CloudFormation 堆疊。

> **Note**
>
> 建立 CloudFormation 堆疊後，您需要支付其佈建的 AWS 資源費用。

使用 AWS CLI 佈建教學課程必要條件

- 執行下列 CLI 命令，其中 `--template-body` 指定範本檔案的位置：

```
aws cloudformation create-stack --stack-name pipe-tuturial-resources --template-
body file://~/pipe-tutorial-resources.json
```

使用 CloudFormation 主控台佈建教學先決條件

1. 在以下網址開啟 AWS CloudFormation 主控台：https://console.aws.amazon.com/cloudformation。
2. 選擇 Stack (堆疊)，然後選取 Create stack (建立堆疊)，選取 With new resources (standard) (使用新資源 (標準))。

   CloudFormation 會顯示建立堆疊精靈。
3. 對於 Prerequisite - Prepare template (先決條件：準備範本)，保持選取 Template is ready (範本已準備就緒)。
4. 針對 Specify template(指定範本)，選取 Upload a template file(上傳範本檔案)，然後選取 Next (下一步)。
5. 設定堆疊及其將佈建的資源：
   - 針對 Stack name (堆疊名稱) 輸入 `pipe-tuturial-resources`。
   - 對於參數，請保留 DynamoDB 資料表和 Amazon SQS 佇列的預設名稱。
   - 選擇 Next (下一步)。
6. 選擇下一步，然後選擇提交。

   CloudFormation 會建立堆疊，並佈建範本中定義的資源。

如需有關 CloudFormation 的詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的什麼是 AWS CloudFormation？。

步驟 1：建立管道

佈建管道來源和目標後，您現在可以建立管道以連接這兩個服務。

使用 EventBridge 控制台建立管道

1. 造訪 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Pipes (管道)。
3. 選擇 Create pipe (建立管道)。
4. 對於名稱，為您的管道 `pipe-tutorial` 命名。
5. 指定 DynamoDB 資料串流來源：

   a. 在詳細資料下，對於來源，選取 DynamoDB 資料串流。

      EventBridge 接會顯示 DynamoDB 特定的來源組態設定。

   b. 對於 DynamoDB 資料串流，請選取 `pipe-tutorial-source`。

      保持起始位置設定為預設值 `Latest`。

   c. 選擇 Next (下一步)。
6. 指定並測試事件模式以篩選事件：

   篩選可讓您控制管道傳送至擴充或目標的事件。管道只會將符合事件模式的事件傳送至擴充或目標。

   如需更多詳細資訊，請參閱 ???。

   > **Note**
   > 您只需為傳送至擴充功能或目標的事件付費。

   a. 在範例事件-選用下，保持選取 AWS 事件，並確定已選取 DynamoDB 串流範例事件 1。

      這是您將用來測試我們的事件模式的示例事件。

   b. 在事件模式下，輸入下列事件模式：

      ```
      {
        "eventName": ["INSERT", "MODIFY"]
      }
      ```

   c. 選擇測試模式。

EventBridge 會顯示一個訊息方塊，說明範例事件符合事件模式。這是因為範例事件的 `eventName` 值為 `INSERT`。

d. 選擇 Next (下一步)。

7. 選擇下一步以略過指定擴充項目。

   在此範例中，您不會選取擴充。擴充可讓您在將來源資料傳送至目標之前，選取服務以增強來源的資料。如需詳細資訊，請參閱 ???。

8. 將您的 Amazon SQS 佇列指定為管道目標：

   a. 在詳細資料下，對於目標服務，選取 Amazon SQS 佇列。

   b. 對於佇列，選取 `pipe-tutorial-target`。

   c. 將目標輸入轉換器區段保留空白。

      如需更多詳細資訊，請參閱 ???。

9. 選擇 Create pipe (建立管道)。

   EventBridge 會建立管道並顯示管路詳細資訊頁面。管的狀態更新為 `Running` 後，即可準備就緒。

步驟 2：確認管道篩選器事件

管道已設置，但尚未從表接收事件。

若要測試管道，您將更新 DynamoDB 表格中的項目。每次更新都會產生 DynamoDB 串流傳送到我們管道的事件。有些會匹配您指定的事件模式，有些則不會。然後，您可以檢查 Amazon SQS 佇列，以確保管道僅交付符合我們事件模式的事件。

更新表格項目以產生事件

1. 請在 https://console.aws.amazon.com/dynamodb/ 開啟 DynamoDB 主控台。
2. 從左側導覽列中，選取 Tables(資料表)。選取 `pipe-tutorial-source` 資料表。

   DynamoDB 會顯示 `pipe-tutorial-source` 的表格詳細資料頁面。

3. 選取瀏覽表格項目，然後選擇建立項目。

   顯示建立項目頁面。

4. 在屬性之下，建立新的表格項目：

a. 針對相簿輸入 `Album A`。

b. 針對 Artist(藝人)，輸入 `Artist A`。

c. 選擇 Create item (建立項目)。

5. 更新表格項目：

a. 在退回的項目下，選擇相簿 A。

b. 選取新增屬性，然後選取字串。

c. 輸入一個新值 Song，其值為 `Song A`。

d. 選擇 Save changes (儲存變更)。

6. 刪除表格項目：

a. 在退回的項目下，檢查相簿 A

b. 在 Actions (動作) 選單中，選取 Delect items (刪除項目)。

您已對表格項目進行了三次更新；這會為 DynamoDB 資料串流產生三個事件：

- 建立項目時的 `INSERT` 事件。
- 當您將屬性新增至項目時的 `MODIFY` 事件。
- 刪除項目時的 `REMOVE` 事件。

但是，您為管道指定的事件模式應該過濾掉任何不是 `INSERT` 或 `MODIFY` 事件的事件。接下來，確認管道是否將預期的事件傳送到佇列。

確認管道是否將預期的事件傳送到佇列。

1. 在 https://console.aws.amazon.com/sqs/ 開啟 Amazon SQS 主控台。
2. 選擇 `pipe-tutorial-target` 佇列。

   Amazon SQS 會顯示佇列詳細資訊頁面。
3. 選取傳送和接收訊息，然後在接收訊息下選擇郵件輪詢。

   佇列會輪詢管道，然後列出其接收的事件。
4. 選擇事件名稱以查看已傳遞的事件 JSON。

隊列中應該有兩個事件：一個事件帶有 INSERT 的 eventName，一個事件帶有 MODIFY 的 eventName。但是，管道並未傳遞刪除表格項目的事件，因為該事件有一個 REMOVE 的 eventName，它與您在管道中指定的事件模式不相符。

步驟 3：清除您的資源

首先，刪除管道本身。

使用 EventBridge 控制台建立管道

1. 造訪 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Pipes (管道)。
3. 選取 pipe-tutorial 管道，然後選擇 Delete (刪除)。

然後，刪除 CloudFormation 堆疊，以避免因其中佈建的資源持續使用量而計費。

使用 AWS CLI 刪除教學課程必要條件

- 執行下列 CLI 命令，其中 --stack-name 指定堆疊的位置：

```
aws cloudformation delete-stack --stack-name pipe-tuturial-resources
```

使用 AWS CloudFormation 主控台刪除教學課程必要條件

1. 在以下網址開啟 AWS CloudFormation 主控台：https://console.aws.amazon.com/cloudformation。
2. 在堆疊頁面上，選取堆疊，然後選取刪除。
3. 選取刪除以確認您的動作。

AWS CloudFormation 模板生成先決條件

使用下面的 JSON 建立 CloudFormation 範本，以佈建本教學課程所需的來源和目標資源。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

 "Description" : "Provisions resources to use with the EventBridge Pipes tutorial. You
 will be billed for the AWS resources used if you create a stack from this template.",
```

```
  "Parameters" : {
    "SourceTableName" : {
      "Type" : "String",
      "Default" : "pipe-tutorial-source",
      "Description" : "Specify the name of the table to provision as the pipe source,
 or accept the default."
    },
  "TargetQueueName" : {
    "Type" : "String",
    "Default" : "pipe-tutorial-target",
    "Description" : "Specify the name of the queue to provision as the pipe target, or
 accept the default."
  }
},
  "Resources": {
    "PipeTutorialSourceDynamoDBTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
        "AttributeDefinitions": [{
            "AttributeName": "Album",
            "AttributeType": "S"
          },
          {
            "AttributeName": "Artist",
            "AttributeType": "S"
          }

        ],
        "KeySchema": [{
            "AttributeName": "Album",
            "KeyType": "HASH"

          },
          {
            "AttributeName": "Artist",
            "KeyType": "RANGE"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 10,
          "WriteCapacityUnits": 10
        },
        "StreamSpecification": {
```

```
          "StreamViewType": "NEW_AND_OLD_IMAGES"
        },
        "TableName": { "Ref" : "SourceTableName" }
      }
    },
    "PipeTutorialTargetQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": { "Ref" : "TargetQueueName" }
      }
    }
  }
}
```

從 EventBridge 管產生 AWS CloudFormation 樣板

AWS CloudFormation 透過將基礎結構視為程式碼，讓您以集中且可重複的方式，跨帳戶和區域設定和管理 AWS 資源。 CloudFormation 透過讓您建立範本 (定義您要佈建和管理的資源) 來執行此作業。

EventBridge 可讓您從帳戶中的現有管道產生範本，以協助您快速開發 CloudFormation 範本。您可以選取要在範本中包含的一或多個管道。然後，您可以使用這些範本作為建立受 CloudFormation 管理資源堆疊的基礎。

若要取得更多資訊 CloudFormation，請參閱《 AWS CloudFormation 使用指南》。

對於事件匯流排，您可以從事件匯流排和事件匯流排規則產生 CloudFormation 範本。

EventBridge 管樣板中包含的資源

EventBridge 產生 CloudFormation 樣板時，會為每個選取的管建立AWS::Pipes::Pipe資源。此外，在描述的條件下 EventBridge 包括以下資源：

- AWS::Events::ApiDestination

  如果您的管道包含 API 目的地 (無論是作為擴充項目或目標)，請將它們作為 AWS::Events::ApiDestination資源 EventBridge包含在 CloudFormation 範本中。

- AWS::Events::EventBus

  如果您的管道包括事件匯流排做為目標，請將其作為 AWS::Events::EventBus 資源 EventBridge 包含在 CloudFormation 樣板中。

- AWS::IAM::Role

  如果您在配置管道時 EventBridge 建立了新的執行角色，則可以選擇將該角色 EventBridge 納入範本中作為 AWS::IAM::Role 資源。 EventBridge 不包括您建立的角色。(在任何一種情況下，AWS::Pipes::Pipe 資源的`RoleArn`屬性都包含角色的 ARN。)

使用從 EventBridge 管產生的 CloudFormation 樣板時的注意事項

使用從中產生的 CloudFormation 範本時，請考量下列因素 EventBridge：

- EventBridge 生成模板中不包含任何密碼。

  您可以編輯範本以包含範本參數，讓使用者在使用範本建立或更新 CloudFormation堆疊時指定密碼或其他敏感資訊。

  此外，使用者可以使用 Secrets Manager 在所需區域中建立密碼，然後編輯產生的範本以使用動態參數。
- 產生之範本中的目標會保持與原始管道中指定的目標完全相同。如果您在使用範本在其他地區建立堆疊之前未適當地編輯範本，這可能會導致跨區域問題。

  此外，產生的範本不會自動建立下游目標。

從 EventBridge 管產生 CloudFormation 樣板

若要使用 EventBridge 控制台從一個或多個管道產生 CloudFormation 樣板，請執行以下操作：

從一個或多個管產生 CloudFormation 樣板的步驟

1. 在 https://console.aws.amazon.com/events/ 打開 Amazon EventBridge 控制台。
2. 在導覽窗格中，選擇管道。
3. 在「管」下，選擇要包括在產生的 CloudFormation 樣板中的一個或多個管。

   對於單一管道，您也可以選擇管道名稱以顯示管道的詳細資訊頁面。
4. 選擇 [CloudFormation 範本]，然後選擇要用 EventBridge 來產生範本的格式：JSON 或 YAML。

   EventBridge 顯示以所選格式產生的樣板。
5. 如果您已為任何選取的管道 EventBridge 建立新的執行角色，並且想 EventBridge 要在範本中包含這些角色，請選擇 [包含由控制台代表您建立的 IAM 角色]。

6. EventBridge 可讓您選擇下載範本檔案，或將範本複製到剪貼簿。

   - 選擇立即下載以下載範本檔案。
   - 若要將範本複製剪貼簿，請選擇複製。

7. 若要結束範本，請選擇取消。

透過全域端點和事件複寫讓應用程式具有區域容錯能力

您可以透過 Amazon EventBridge 全球端點提高應用程式的可用性。全域端點可讓應用程式區域容錯而無需增加額外費用。若要開始，請將 Amazon Route 53 運作狀態檢查指派給端點。啟動容錯移轉時，運作狀態檢查會報告「狀況不良」狀態。在容錯移轉初始化的幾分鐘內，所有自訂事件都會路由至次要區域中的事件匯流排，並由該事件匯流排處理。一旦運作狀態檢查報告為「狀況良好」狀態，事件就會由主要區域中的事件匯流排處理。

當您使用全域端點時，您可以啟用事件複寫。事件複寫會使用受管規則，將所有自訂事件傳送至主要和次要區域中的事件匯流排。

> **Note**
>
> 如果您使用自訂匯流排，則需要在每個區域中使用相同名稱且相同帳戶中的自訂匯流排，以便容錯移轉正常運作。

主題

復原時間與復原點目標

復原時間點目標 (RTO) 是次要區域在失敗後開始接收事件所需的時間。對於 RTO，此時間包括觸發 CloudWatch 警示和更新 Route 53 運作狀態檢查狀態的時間段。復原點目標 (RPO) 是在失敗期間保持未處理之資料的度量。對於 RPO，時間包括未複製到次要區域並停留在主要區域中的事件，直到服務或區域復原為止。使用全球端點時，如果您遵循我們的警示設定規範指引，您可以預期 RTO 和 RPO 為 360 秒，最長 420 秒。

複寫事件

在次要區域中以非同步方式處理事件。這表示不能保證在兩個區域中同時處理事件。觸發容錯移轉時，事件會由次要區域處理，並在主要區域可用時由主要區域進行處理。啟用事件複寫會增加您的每月成本。如需詳細資訊，請參閱 Amazon EventBridge 定價。

建議您在設定全域端點時啟用事件複寫，原因如下：

- 事件複寫可協助您確認已正確設定全域端點。這有助於確保您在容錯移轉的情況下得到保障。
- 需要事件複寫，才能從容錯移轉事件自動復原。如果您沒有啟用事件複寫，您必須手動將 Route 53 運作狀態檢查重設為「狀況良好」，事件才會回到主要區域。

複製的事件承載

以下是複製的事件承載的範例。

> **Note**
>
> 針對 `region`，會列出事件複製來源的「區域」。

```
{
    "version": "0",
    "id": "a908baa3-65e5-ab77-367e-527c0e71bbc2",
    "detail-type": "Test",
    "source": "test.service.com",
    "account": "0123456789",
    "time": "1900-01-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:events:us-east-1:0123456789:endpoint/MyEndpoint"
    ],
    "detail": {
        "a": "b"
    }
}
```

建立全球端點

請完成下列步驟來設定全域端點：

1. 請確定您在主要和次要區域中都有相符的事件匯流排和規則。
2. 建立 Route 53 運作狀態檢查以監控您的事件匯流排。如需建立運作狀態檢查的協助，請在建立全域端點時選擇新增健康狀態檢查。
3. 建立您的全球端點

設定 Route 53 運作狀態檢查之後，您就可以建立全域端點。

使用主控台建立全域端點

1. 造訪 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 全球端點。
3. 選擇 Create Endpoint (建立端點)。
4. 輸入端點的名稱和描述。
5. 對於主要區域中的事件匯流排，請選擇您想要與端點相關聯的事件匯流排。
6. 對於次要區域，請選擇發生容錯移轉時要將事件引導至的區域。

> **Note**
>
> 次要區域中的事件匯流排會自動填入且無法編輯。

7. 對於 Route 53 運作狀態檢查以進行觸發容錯移轉和復原，請選擇端點將監控的健全狀況檢查。如果您尚未進行運作狀態檢查，請選擇新增健全狀況檢查以開啟 AWS CloudFormation 主控台，並使用 CloudFormation 範本建立運作狀態檢查。

> **Note**
>
> 遺失資料會導致運作狀態檢查失敗。如果您只需要間歇性地傳送事件，請考慮使用較長的最小值週期，或將遺失的資料視為「遺失」而非「違規」。

8. (可選) 對於事件複製，請執行下列動作：

   a. 選取已啟用事件複製。

b. 對於執行角色,選擇建立新的 AWS Identity and Access Management 角色，或使用現有的角色。請執行下列動作：

- 選擇 Create a new role for this specific resource (為此特定資源建立新角色)。或者，您可以更新角色名稱以建立新角色。
- 否則，請選擇 Use existing role (使用現有角色)。然後，對於執行角色，選擇要使用的所需角色。

9. 選擇 Create (建立)。

使用主控台建立全域端點

若要使用 EventBridge API 建立全球端點，請參閱《Amazon EventBridge API 參考》中的 CreateEndpoint。

使用 AWS CloudFormation 建立全域端點

若要使用 AWS CloudFormation API 建立全域端點，請參閱《AWS CloudFormation 使用者指南中的》中的 AWS::Events::Endpoints。

使用 AWS SDK 使用全域端點

> **Note**
>
> 即將推出 C++ 支援。

當使用 AWS SDK 與全域端點搭配使用時，請記住下列事項：

- 您需要為您的特定 SDK 安裝 AWS 通用執行期 (CRT) 庫。如果您沒有安裝 CRT，您會看到一則例外狀況訊息，指出需要安裝的項目。如需詳細資訊，請參閱下列內容：
  - AWS 通用執行期 (CRT) 程式庫
  - awslabs/aws-crt-java
  - awslabs/aws-crt-nodejs
  - awslabs/aws-crt-python
- 建立全域端點之後，您需要將 `endpointId` 和 `EventBusName` 新增至您使用的任何 `PutEvents` 呼叫中。

- 全域端點支援簽章版本 4A。這個版本的 SigV4 允許為多個 AWS 區域 簽署請求。這對於可能導致來自數個區域之一的資料存取的 API 操作非常有用。當您使用 AWS SDK 時，您提供您的憑證，而對多區域端點的請求將使用簽章版本 4A，而不需要其他設定。如需 SigV4A 的詳細資訊，請參閱《AWS 一般參考》中的簽署 AWS API 請求。

可用的區域

以下區域支援端點。

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 南美洲 (聖保羅)

使用 Amazon EventBridge 全球端點的最佳實務

設定全域端點時，建議採用下列最佳作法。

主題

- 啟用事件複寫
- 防止事件限流
- 使用 Amazon Route 53 運作狀態檢查中的訂閱用戶指標

啟用事件複寫

強烈建議您在指派給全域端點的次要區域中開啟複寫並處理事件。這可確保您在次要區域中的應用程式設定正確。您也應該開啟複寫功能，以確保在緩解問題後自動復原至主要區域。

事件 ID 可能會隨 API 呼叫而變更，因此跨區域的事件關聯需要您具有不可變的唯一識別碼。消費者也應該考慮等冪性的設計。如此一來，如果您要複製事件，或從封存中重新播放事件，兩個區域中處理的事件就不會產生任何副作用。

防止事件限流

為了防止事件受到限制，我們建議您更新 `PutEvents` 和目標限制，使其在不同區域之間保持一致。

使用 Amazon Route 53 運作狀態檢查中的訂閱用戶指標

避免包含 Amazon Route 53 運作狀態檢查中的訂閱用戶指標 如果訂閱用戶遇到問題，儘管主要區域中的所有其他訂閱用戶都保持良好狀態，但包括這些指標可能會導致您的發布者容錯移轉到次要區域。如果您的其中一個訂閱用戶無法處理主要區域中的事件，您應該開啟複寫功能，以確保次要區域的訂閱用戶能夠順利處理事件。

AWS CloudFormation 設定 Route 53 運作狀態檢查的範本

使用全域端點時，您必須進行 Route 53 運作狀態檢查才能監控區域的狀態。下列範本定義了一個Amazon CloudWatch 警示，並使用它來定義Route 53 運作狀態檢查。

主題

- 用於定義 Route 53 運作狀態檢查的 AWS CloudFormation 範本
- CloudWatch 警示範本屬性
- 路由 53 運作狀態檢查範本屬性

用於定義 Route 53 運作狀態檢查的 AWS CloudFormation 範本

使用下列範本來定義 Route 53 運作狀態檢查。

```
Description: |-
  Global endpoints health check that will fail when the average Amazon EventBridge
  latency is above 30 seconds for a duration of 5 minutes. Note, missing data will
  cause the health check to fail, so if you only send events intermittently, consider
  changing the heath check to use a longer evaluation period or instead treat missing
  data as 'missing' instead of 'breaching'.

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Global endpoint health check alarm configuration"
        Parameters:
          - HealthCheckName
          - HighLatencyAlarmPeriod
          - MinimumEvaluationPeriod
          - MinimumThreshold
          - TreatMissingDataAs
    ParameterLabels:
      HealthCheckName:
        default: Health check name
      HighLatencyAlarmPeriod:
        default: High latency alarm period
      MinimumEvaluationPeriod:
        default: Minimum evaluation period
      MinimumThreshold:
        default: Minimum threshold
      TreatMissingDataAs:
        default: Treat missing data as

Parameters:
  HealthCheckName:
    Description: Name of the health check
    Type: String
    Default: LatencyFailuresHealthCheck
  HighLatencyAlarmPeriod:
    Description: The period, in seconds, over which the statistic is applied. Valid
 values are 10, 30, 60, and any multiple of 60.
    MinValue: 10
    Type: Number
    Default: 60
  MinimumEvaluationPeriod:
```

```
    Description: The number of periods over which data is compared to the specified
 threshold. You must have at least one evaluation period.
    MinValue: 1
    Type: Number
    Default: 5
  MinimumThreshold:
    Description: The value to compare with the specified statistic.
    Type: Number
    Default: 30000
  TreatMissingDataAs:
    Description: Sets how this alarm is to handle missing data points.
    Type: String
    AllowedValues:
      - breaching
      - notBreaching
      - ignore
      - missing
    Default: breaching

Mappings:
  "InsufficientDataMap":
    "missing":
      "HCConfig": "LastKnownStatus"
    "breaching":
      "HCConfig": "Unhealthy"

Resources:
  HighLatencyAlarm:
      Type: AWS::CloudWatch::Alarm
      Properties:
        AlarmDescription: High Latency in Amazon EventBridge
        MetricName: IngestionToInvocationStartLatency
        Namespace: AWS/Events
        Statistic: Average
        Period: !Ref HighLatencyAlarmPeriod
        EvaluationPeriods: !Ref MinimumEvaluationPeriod
        Threshold: !Ref MinimumThreshold
        ComparisonOperator: GreaterThanThreshold
        TreatMissingData: !Ref TreatMissingDataAs

  LatencyHealthCheck:
      Type: AWS::Route53::HealthCheck
      Properties:
        HealthCheckTags:
```

```
          - Key: Name
            Value: !Ref HealthCheckName
        HealthCheckConfig:
          Type: CLOUDWATCH_METRIC
          AlarmIdentifier:
            Name:
              Ref: HighLatencyAlarm
            Region: !Ref AWS::Region
          InsufficientDataHealthStatus: !FindInMap [InsufficientDataMap, !Ref
 TreatMissingDataAs, HCConfig]

Outputs:
  HealthCheckId:
    Description: The identifier that Amazon Route 53 assigned to the health check when
 you created it.
    Value: !GetAtt LatencyHealthCheck.HealthCheckId
```

事件 ID 可能會隨 API 呼叫而變更，因此跨區域的事件關聯需要您具有不可變的唯一識別碼。消費者也應該考慮等冪性的設計。如此一來，如果您要複製事件，或從封存中重新播放事件，兩個區域中處理的事件就不會產生任何副作用。

CloudWatch 警示範本屬性

> **Note**
>
> 對於所有 **`editable`** 欄位，請考慮每秒輸送量。如果您只是間歇性地傳送事件，請考慮將健康檢查變更為使用較長的評估期間，或者改為將遺失的資料視為 `missing` 而非 `breaching`。

下列屬性用於範本的 CloudWatch 警示區段：

指標	描述
`AlarmDescription`	警示的說明。 預設：**`High Latency in Amazon EventBridge`**
`MetricName`	與警示相關聯的指標名稱。這對以指標為基礎的警示是必要的。針對以數學運算式為基礎的警示，您要改用 `Metrics`，而且不能指定 `MetricName` 。

指標	描述
	預設值：IngestionToInvocationStartLatency
Namespace	與警示相關聯之指標的命名空間。這對以指標為基礎的警示是必要的。針對以數學運算式為基礎的警示，您不能指定 Namespace ，要改用 Metrics。 預設：AWS/Events
Statistic	除百分位數外，與警示相關聯的指標統計。 預設值：平均值
Period	套用統計資料的期間 (以秒為單位)。這對以指標為基礎的警示是必要的。有效值為 10、30、60，以及 60 的任何倍數。 預設：**60**
EvaluationPeriods	執行資料和指定閾值比較作業的週期。如果您設定的警示需連續違反數個資料點才能觸發警示，則此值會指定該數目。如果您要設定「N 個中有 M 個」的警示，則此值為 N，且 DatapointsToAlarm 為 M。 預設：**5**
Threshold	要與指定統計資料比較的值。 預設：**30,000**
ComparisonOperator	與指定的統計資料和閾值比較時，要使用的算術操作。指定的統計值會作為第一個運算元使用。 預設：GreaterThanThreshold
TreatMissingData	設定此警示處理缺失資料點的方式。 有效值：breaching 、notBreaching 、ignore 和 missing 預設：breaching

路由 53 運作狀態檢查範本屬性

> **Note**
>
> 對於所有 **editable** 欄位，請考慮每秒輸送量。如果您只是間歇性地傳送事件，請考慮將健康檢查變更為使用較長的評估期間，或者改為將遺失的資料視為 missing 而非 breaching。

下列屬性用於範本的 Route 53 運作狀態檢查區段：

指標	描述
HealthCheckName	運作狀態檢查的名稱。 預設：**LatencyFailuresHealthCheck**
InsufficientDataHealthStatus	當 CloudWatch 用來判斷警示狀態的指標資料不足時，您想要 Amazon Route 53 指派給運作狀態檢查的狀態： 有效值： • Healthy：Route 53 將這些運作狀態檢查視為正常。 • Unhealthy ：Route 53 將這些運作狀態檢查視為狀況不良。 • LastKnownStatus ：Route 53 使用上次 CloudWatch 有足夠資料時的運作狀態檢查狀態，判斷警示狀態。對於沒有上次已知狀態的新運作狀態檢查，運作狀態檢查的預設狀態是正常。 預設值：狀態不良 **Note** 此欄位會根據對 TreatMissingData 欄位的輸入進行更新。如果設定 TreatingMissingData 為 Missing，則會將其更新為 LastKnownStatus 。如果 TreatingMissingData 設定為 Breaching ，則會將其更新為 Unhealthy 。

Amazon EventBridge 模式

結構描述定義傳送至的事件結構 EventBridge。 EventBridge 為 AWS 服務產生的所有事件提供結構描述。您也可以建立或上傳結構描述，或直接從事件匯流排上的事件推斷結構描述。擁有事件的結構描述後，您就可以下載常用程式設計語言的程式碼繫結以及加速開發。您可以使用結構描述的程式碼繫結，並從 EventBridge 主控台、使用 API 或直接在 IDE 中使用工 AWS 具組來管理結構描述。若要建置使用事件的無伺服器應用程式，請使用 AWS Serverless Application Model。

> **Note**
>
> 使用輸入轉換器功能時，結構描述探索會推斷原始事件，而不是傳送至目標的轉換事件。

EventBridge 同時支持 OpenAPI 3 和 JSON 模式草稿 4 格式。

對於AWSVS Code 的AWS 工具組 JetBrains和工具組，您可以瀏覽或搜尋結構描述，並直接在 IDE 中下載結構描述的程式碼繫結。

下列影片提供結構描述和結構描述登錄檔的概觀：使用結構描述登錄檔

主題

- 結構描述登錄檔 API 屬性值遮罩
- 查找 Amazon EventBridge 模式
- Amazon EventBridge 模式註冊表
- 創建一個 Amazon EventBridge 模式
- Amazon EventBridge 代碼綁定

結構描述登錄檔 API 屬性值遮罩

用來建立結構描述登錄檔之事件的某些屬性值可能包含敏感的客戶資訊。為了保護客戶的資訊，數值將以星號 (*) 遮罩。因為我們要遮罩這些值，所以 EventBridge 建議您不要建置明確依賴下列屬性或其值的應用程式：

- CreateSchema— 身`requestParameters`體的`Content`屬性

- GetDiscoveredSchema— `requestParameters` 身體的`Events`屬性和身`responseElements`體的`Content`屬性
- SearchSchemas— 的`keywords`財產 `requestParameters`
- UpdateSchema— 的`Content`財產 `requestParameters`

查找 Amazon EventBridge 模式

EventBridge 包括產生事件的所有 AWS 服務的結構描述。您可以在 EventBridge 主控台中找到這些結構描述，也可以使用 API 動作尋找結構描述SearchSchemas。

在 EventBridge 主控台中尋找 AWS 服務的結構描述

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述。
3. 在結構描述頁面上，選取 AWS 事件結構描述登錄檔。
   <result>

   可用結構描述的第一頁將會顯示。
   </result>
4. 若要尋找結構描述，請在搜尋 AWS 事件結構描述中輸入搜尋字詞。

   搜尋會傳回與可用結構描述名稱和內容相符的項目，並顯示找到該結構描述的版本。
5. 選取結構描述的名稱以開啟事件結構描述。

Amazon EventBridge 模式註冊表

結構描述登錄檔是結構描述的容器。結構描述登錄檔會收集並組織結構描述，讓您的結構描述位於邏輯群組中。預設結構描述登錄檔如下：

- 所有結構描述 — AWS 事件、探索到的和自訂結構描述登錄中的所有結構描述。
- AWS 事件結構描述登錄 — 內建結構描述。
- 探索到的結構描述登錄檔：結構描述探索所發現的結構描述。

您可以建立自訂登錄檔，以組織您建立或上傳的結構描述。

建立自訂登錄檔

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述，然後選擇建立登錄檔。
3. 在登錄檔詳細資訊頁面上，輸入名稱。
4. (選用) 輸入新登錄檔的描述。
5. 選擇建立。

若要在新登錄檔中建立自訂結構描述，請選取建立自訂結構描述。若要將結構描述新增至登錄檔，請在建立新結構描述時選取該登錄檔。

若要使用 API 建立登錄檔，請使用 `CreateRegistry`。如需詳細資訊，請參閱 Amazon EventBridge 架構登錄檔 API 參考。

如需透過使用 EventBridge 結構描述登錄的詳細資訊 AWS CloudFormation，請參閱中的EventSchemas 資源類型參考 AWS CloudFormation。

創建一個 Amazon EventBridge 模式

您可以透過使用 JSON 檔案與 OpenAPI 規範或 JSONSchema Draft4 規範建立結構描述。您可以在 EventBridge 中建立或上傳自己的結構描述，方法是使用範本或根據事件的 JSON 產生結構定義。您也可以從事件匯流排上的事件推斷結構描述。若要使用結構描述登錄檔 API 建立 EventBridge 結構描述，請使用 `CreateSchema`API 動作。

當您在 OpenAPI 3 和 JSONSchema Draft4 格式之間進行選擇時，請考慮以下差異：

- JSONSchema 格式支援 OpenAPI 中不支援的其他關鍵字，例如 `$schema, additionalItems`。
- 關鍵字的處理方式有些微差異，例如 `type` 和 `format`。
- OpenAPI 不支援 JSON 文件中的 JSONSchema 超結構描述超連結。
- OpenAPI 的工具傾向於專注於構建時，而 JSONSchema 的工具傾向於專注於運行時操作，例如用於模式驗證的客戶端工具。

我們建議使用 JsonSchema 格式來實現客戶端驗證，以便發送的事件 EventBridge 符合模式。您可以使用 JSONSchema 來定義有效 JSON 文件的合約，然後在傳送相關事件之前使用 JSON 結構描述驗證程式。

建立新結構描述之後，您可以下載程式碼繫結，以協助建立具有該結構描述之事件的應用程式。

主題

- 使用範本建立結構描述
- 直接在主控台中編輯結構描述範本
- 從事件的 JSON 建立結構描述
- 從事件匯流排上的事件建立結構描述

使用範本建立結構描述

您可以從範本建立結構描述，或直接在 EventBridge主控台中編輯範本。若要獲取範本，請從主控台下載範本。您可以編輯範本，使結構描述符合您的事件。然後透過主控台上傳新範本。

下載結構描述範本

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述登錄檔。

3. 在結構描述範本下的入門區段中，選擇下載。

或者，您可以從下列程式碼範例下載 JSON 範本。

```
{
    "openapi": "3.0.0",
    "info": {
      "version": "1.0.0",
      "title": "Event"
    },
    "paths": {},
    "components": {
      "schemas": {
        "Event": {
          "type": "object",
          "properties": {
            "ordinal": {
              "type": "number",
              "format": "int64"
            },
            "name": {
              "type": "string"
            },
            "price": {
              "type": "number",
              "format": "double"
            },
            "address": {
              "type": "string"
            },
            "comments": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "created_at": {
              "type": "string",
              "format": "date-time"
            }
          }
        }
      }
```

```
    }
  }
```

上傳結構描述範本

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述，然後選擇建立結構描述。
3. (選用) 選取或建立結構描述登錄檔。
4. 在結構描述詳細資訊下，輸入結構描述的名稱。
5. (選用) 輸入結構描述的說明。
6. 針對結構描述類型，請選擇 OpenAPI 3.0 或 JSON 結構描述草稿 4。
7. 在文字方塊的建立標籤上，將結構描述檔案拖曳至文字方塊，或貼上結構描述來源。
8. 選取建立。

直接在主控台中編輯結構描述範本

在主控台中編輯結構描述

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述，然後選擇建立結構描述。
3. (選用) 選取或建立結構描述登錄檔。
4. 在結構描述詳細資訊下，輸入結構描述的名稱。
5. 針對結構描述類型，請選擇 OpenAPI 3.0 或 JSON 結構描述草稿 4。
6. (選用) 為您建立的結構描述輸入描述。
7. 在建立標籤上，選擇載入範本。
8. 在文字方塊內編輯範本，使結構描述符合您的事件。
9. 選取建立。

從事件的 JSON 建立結構描述

如果您擁有事件的 JSON，則可以自動為該類型的事件建立結構描述。

根據事件的 JSON 建立結構描述

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述，然後選擇建立結構描述。
3. (選用) 選取或建立結構描述登錄檔。
4. 在結構描述詳細資訊下，輸入結構描述的名稱。
5. (選用) 為您建立的結構描述輸入描述。
6. 針對結構描述類型，請選擇 OpenAPI 3.0。

   當您從事件的 JSON 建立結構定義時，您無法使用 JSONSchema。
7. 選取從 JSON 探索
8. 在 JSON 下的文字方塊中，貼上或拖曳事件的 JSON 來源。

   例如，您可以貼上來自此 AWS Step Functions 事件的來源，以便執行失敗。

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "012345678912",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-1",
    "resources": [
      "arn:aws:states:us-east-1:012345678912:execution:state-machine-
name:execution-name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-1:012345678912:execution:state-
machine-name:execution-name",
        "stateMachineArn": "arn:aws:states:us-
east-1:012345678912:stateMachine:state-machine",
        "name": "execution-name",
        "status": "FAILED",
        "startDate": 1551225146847,
        "stopDate": 1551225151881,
        "input": "{}",
        "output": null
    }
}
```

9. 選擇探索結構描述。
10. EventBridge 會產生事件的 OpenAPI 結構描述。例如，會針對先前的 Step Functions 事件產生下列結構描述。

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "1.0.0",
    "title": "StepFunctionsExecutionStatusChange"
  },
  "paths": {},
  "components": {
    "schemas": {
      "AWSEvent": {
        "type": "object",
        "required": ["detail-type", "resources", "detail", "id", "source", "time",
 "region", "version", "account"],
        "x-amazon-events-detail-type": "Step Functions Execution Status Change",
        "x-amazon-events-source": "aws.states",
        "properties": {
          "detail": {
            "$ref": "#/components/schemas/StepFunctionsExecutionStatusChange"
          },
          "account": {
            "type": "string"
          },
          "detail-type": {
            "type": "string"
          },
          "id": {
            "type": "string"
          },
          "region": {
            "type": "string"
          },
          "resources": {
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          "source": {
            "type": "string"
```

```
          },
          "time": {
            "type": "string",
            "format": "date-time"
          },
          "version": {
            "type": "string"
          }
        }
      },
      "StepFunctionsExecutionStatusChange": {
        "type": "object",
        "required": ["output", "input", "executionArn", "name", "stateMachineArn",
"startDate", "stopDate", "status"],
        "properties": {
          "executionArn": {
            "type": "string"
          },
          "input": {
            "type": "string"
          },
          "name": {
            "type": "string"
          },
          "output": {},
          "startDate": {
            "type": "integer",
            "format": "int64"
          },
          "stateMachineArn": {
            "type": "string"
          },
          "status": {
            "type": "string"
          },
          "stopDate": {
            "type": "integer",
            "format": "int64"
          }
        }
      }
    }
  }
```

```
}
```

11. 產生結構描述之後，請選擇建立。

從事件匯流排上的事件建立結構描述

EventBridge 可以透過探索事件來推斷結構描述。若要推斷結構描述，您可以在事件匯流排上開啟事件探索，並將每個唯一的結構描述新增至結構描述登錄檔，包括跨帳戶事件的結構描述。發現的綱要 EventBridge 會顯示在「結構描述」頁面的「探索到的綱要」登

如果事件匯流排上的事件內容發生變更，則 EventBridge 會建立相關 EventBridge 結構描述的新版本。

> **Note**
>
> 在事件匯流排上啟用事件探索可能會產生成本。每個月的前五百萬個處理事件是免費的。

> **Note**
>
> EventBridge 依預設，會從跨帳戶事件推斷結構描述，但您可以透過更新屬`cross-account`性來停用結構描述。如需詳細資訊，請參閱 EventBridge 結構描述登錄 API 參照中的探索器。

啟用事件匯流排上的結構描述探索

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇事件匯流排。
3. 執行以下任意一項：
   - 若要在預設事件匯流排上啟用探索，請選擇開始探索。
   - 若要在自訂事件匯流排上啟用探索，請選取自訂事件匯流排的選項按鈕，然後選擇開始探索。

Amazon EventBridge 代碼綁定

您可以產生事件結構描述的程式碼繫結，以加快 Golang、Java、Python 和 TypeScript. 程式碼繫結可用於 AWS 服務事件、您建立的結構描述，以及根據事件匯流排上的事件產生的結構描述。您可以使用 EventBridge 主控台、結構描述登錄 API，或在 IDE 中使用 AWS 工具組來產生結 EventBridge 構描述的程式碼繫結。

若 EventBridge 要從結構描述產生程式碼繫結

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇結構描述。
3. 透過瀏覽結構描述登錄檔或搜尋結構描述，尋找您想要程式碼繫結的結構描述。
4. 選取結構描述名稱。
5. 在版本區段中的結構描述詳細資訊頁面，選擇下載程式碼繫結。
6. 在下載程式碼繫結頁面上，選取您要下載的程式碼繫結語言。
7. 選取下載。

   開始下載可能需要幾秒鐘的時間。下載的檔案將是您所選取語言的程式碼繫結 zip 檔案。

Amazon EventBridge 相關服務和工具

Amazon EventBridge 可與其他 AWS 服務搭配使用，以處理事件或調用資源作為規則的目標。如需 EventBridge 和其他 AWS 服務和工具整合的詳細資訊，請參閱下列內容：

主題

- 使用 Amazon EventBridge 搭配界面 VPC 端點
- Amazon EventBridge 與 AWS X-Ray 整合
- 搭 EventBridge 配 AWS 整合式應用測試套件使用
- 在 AWS CloudFormation 堆疊中包括 Amazon EventBridge 資源

使用 Amazon EventBridge 搭配界面 VPC 端點

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 來託管您的 AWS 資源，可以在您的 VPC 與 EventBridge 之間建立連線。VPC 上的資源可以使用此連線來與 EventBridge通訊。

您可利用 VPC 來控制您的網路設定，例如 IP 地址範圍、子網路、路由表和網路閘道。若要將您的 VPC 連接到 EventBridge，請定義 EventBridge 的 VPC 端點界面。端點能為 EventBridge 提供可靠、可擴展性的連線，無須使用網際網路閘道、網路位址轉譯 (NAT) 執行個體或 VPN 連接。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC。

VPC 端點介面使用 AWS PrivateLink，讓私有通訊使用於彈性網路界面與 AWS 服務之間的私有 IP 地址。如需詳細資訊，請參閱 AWS PrivateLink 和 VPC 端點。

當您使用私有介面虛擬私人 VPS 端點時，您的 VPC 傳送給 EventBridge 的自訂事件會使用該端點。然後，EventBridge 會根據您設定的規則和目標，將這些事件傳送至其他 AWS 服務。將事件傳送至其他服務後，您可以透過該服務的公用端點或 VPC 端點接收事件。例如，如果您建立規則將事件傳送到 Amazon SQS 佇列，您可以為 Amazon SQS 設定介面 VPC 端點，以接收來自 VPC 中該佇列的訊息，而無需使用公有端點。

可用性

EventBridge 目前在下列區域支援 VPC 端點：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太區域 (孟買)
- 亞太區域 (海德拉巴)
- 亞太區域 (香港)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (雅加達)
- 亞太區域 (墨爾本)
- 亞太區域 (東京)

- 亞太區域 (首爾)
- 亞太區域 (大阪)
- 加拿大 (中部)
- 加拿大西部 (卡加利)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (蘇黎世)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- 歐洲 (西班牙)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 中東 (阿拉伯聯合大公國)
- 中東 (巴林)
- 南美洲 (聖保羅)
- 以色列 (特拉維夫)
- AWS GovCloud (美國西部)
- AWS GovCloud (美國東部)

為 EventBridge 建立 VPC 端點

若要在 VPC 中使用 EventBridge，請為 EventBridge建立界面 VPC 端點，並選擇 com.amazonaws.***Region***.events 作為服務名稱。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立界面端點。

EventBridge 管道細節

無法使用介面 VPC 端點的完整 EventBridge 管道支援。若要在 VPC 內搭配 EventBridge 管道使用下列來源，請參閱下列內容：

- Amazon MSK 網路組態

- 自我管理的 Apache Kafka 網路組態
- Amazon MQ 網路組態

Amazon EventBridge 與 AWS X-Ray 整合

您可以使用 AWS X-Ray 來追蹤通過 EventBridge 的事件。EventBridge 會將原始追蹤標頭傳遞至目標，以便目標服務可以追蹤、分析和偵錯。

只有當事件來自傳遞追蹤內容的 `PutEvents` 要求時，EventBridge 才能傳遞事件的追蹤標頭。X-Ray 不會追蹤來自第三方合作夥伴、排程事件或AWS 服務的事件，而且這些事件來源不會顯示在 X-Ray 服務地圖上。

X-Ray 會驗證追蹤標頭，而且會捨棄無效的追蹤標頭。但是，仍會處理該事件。

> **Important**
>
> 追蹤標頭在傳遞至調用目標的事件上無法使用。
>
> - 如果您有事件封存，則已封存的事件上無法使用追蹤標頭。如果您重新顯示封存的事件，則不會包含追蹤標頭。
> - 如果您有無效字母佇列 (DLQ)，追蹤標頭會包含在將事件傳送至 DLQ 的 `SendMessage` 要求中。如果您使用 `ReceiveMessage` 從 DLQ 擷取事件 (訊息)，與事件相關聯的追蹤標頭會包含在 Amazon SQS 訊息屬性中，但事件訊息中不會包含該標頭。

如需 EventBridge 事件節點如何連接來源和目標服務的詳細資訊，請參閱 《AWS X-Ray 開發人員指南》中的在 X-Ray 服務對應中檢視來源和目標。

您可以透過 EventBridge 傳遞下列追蹤標頭資訊：

- 預設 HTTP 標頭：X-Ray SDK 會自動填入追蹤標頭做為所有調用目標的 `X-Amzn-Trace-Id` HTTP 標頭。若要進一步了解預設 HTTP 標頭，請參閱《AWS X-Ray 開發人員指南》中的追蹤標頭。
- **`TraceHeader`系統屬性**：`TraceHeader` 是由EventBridge 保留的 PutEventsRequestEntry 屬性，可將 X-Ray 追蹤標頭傳送至目標。如果您也使用 `PutEventsRequestEntry`，則 `PutEventsRequestEntry` 會覆寫 HTTP 追蹤標頭。

> **Note**
>
> 追蹤標頭不會計入 `PutEventsRequestEntry` 事件大小。如需更多詳細資訊，請參閱 計算 Amazon EventBridge PutEvents 事件條目大小。

下面為演示 X-Ray 和 EventBridge 一起使用的視頻：使用 AWS X-Ray 來追蹤

搭 EventBridge 配 AWS 整合式應用測試套件使用

當您建立由 Lambda 或 Step Functions 等無伺服器服務組成的應用程式時，您的許多架構元件無法部署到桌面，而是僅存在於 AWS 雲端中。 EventBridge與使用本機部署的應用程式不同，這些類型的應用程式會受益於執行自動化測試的雲端策略。 AWS 整合式應用程式測試套件 (AWS IATK) 可協助您針對應用程式實作其中一些策略。

AWS IATK 是一個軟件庫，可幫助您為基於雲的應用程序編寫自動化測試。

EventBridge 與 AWS IATK 的整合

您可以搭配 AWS IATK 使用 EventBridge 事件和事件匯流排來實作自動化測試，包括：

實作測試線束

若要撰寫事件驅動架構的整合測試，請將應用程式分解為子系統，以建立邏輯界限。測試子系統的一個有用技術是建立測試線束；也就是說，您專門為測試子系統而建立的資源。

例如，整合測試可以透過將輸入測試事件傳遞給子系統程序來開始子系統處理程序。 AWS IATK 可以為您建立偵聽輸出事件的測試工具。 EventBridge (在引擎蓋下，線束是由將輸出事件轉寄至 Amazon SQS 的 EventBridge 規則所組成。) 然後，您的集成測試查詢測試工具以檢查輸出並確定測試是否通過或失敗。

產生模擬事件

AWS IATK 可讓您從結構描述登錄檔中儲存的結構描述產生模擬事件。 EventBridge 這可讓您產生模擬事件，並使用產生的事件調用任何取用者 (例如 Lambda 函數或 Step Functions 函數狀態機器)。

如需詳細資訊，請參閱上的AWS 整合應用程式測試套件概觀 GitHub。

在 AWS CloudFormation 堆疊中包括 Amazon EventBridge 資源

AWS CloudFormation 透過將基礎架構視為程式碼，讓您以集中且可重複的方式，跨帳戶和區域設定和管理 AWS 資源。CloudFormation 透過讓您建立範本 (定義您要佈建和管理的資源) 來執行此作業。這

些資源可能包括 EventBridge 成品，例如事件匯流排和規則、管道、結構描述和排程等。使用這些資源，以便將 EventBridge 功能包含在您透過 CloudFormation 佈建和管理的技術堆疊中。

在 AWS CloudFormation 中可用的 Amazon EventBridge 資源

EventBridge 在下列資源命名空間中提供可用於 CloudFormation 範本的資源：

- AWS::Events

  範本範例包括：

  - 為 PagerDuty 建立 API 目的地
  - 為 Slack 建立 API 目的地
  - 使用 ApiKey 授權參數建立連線
  - 使用 OAuth 授權參數建立連線
  - 使用事件複寫建立全域端點
  - 使用多個主體和動作的拒絕政策
  - 使用自訂事件匯流排將許可授予組織
  - 建立跨區域規則
  - 為目標建立內含無效字母佇列的規則
  - 定期調用 Lambda 函數
  - 調用 Lambda 函數以回應事件
  - 通知主題以回應日誌項目
- AWS::EventSchemas
- AWS::Pipes

  範本範例包括：

  - 使用事件篩選條件建立管道
- AWS::Scheduler

為 AWS CloudFormation 範本產生 Amazon EventBridge 資源定義

EventBridge 主控台可讓您從帳戶中的現有事件匯流排、規則和管道建立 CloudFormation 範本，協助您快速開始開發 CloudFormation 範本。

- ???

- ???
- ???

使用 EventBridge 管理 AWS CloudFormation 堆疊事件

除了在 CloudFormation 堆疊中包含 EventBridge 資源之外，您還可以使用 EventBridge 來管理由 CloudFormation 堆疊本身所產生的事件。每當對堆疊執行建立、更新、刪除或漂移偵測作業時，CloudFormation 都會將事件傳送到 EventBridge。CloudFormation 也會針對堆疊集和堆疊集執行個體的狀態變更，將事件傳送至 EventBridge。您可以使用 EventBridge 規則將事件路由至您定義的目標。

如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的使用 EventBridge 管理 CloudFormation 事件。

Amazon EventBridge 教學課程

將 EventBridge 與多個 AWS 服務和 SaaS 合作夥伴整合。這些教學課程旨在協助您熟悉 EventBridge 的基礎知識及其如何成為無伺服器架構的一部分。

教學：

- Amazon EventBridge 入門教學課程
- 與其他 AWS 服務整合的 Amazon EventBridge 教學課程
- 與 SaaS 供應商整合的 Amazon EventBridge 教學課程

Amazon EventBridge 入門教學課程

下列教學課程可協助您探索 EventBridge 的功能及其使用方法。

教學：

- 封存與重播 Amazon EventBridge 事件
- 建立 Amazon EventBridge 範例應用程式
- 教學課程：使用 EventBridge 結構描述登錄檔下載事件的程式碼繫結
- 教學課程：使用輸入轉換器以自訂 EventBridge 傳送至事件目標的內容

封存與重播 Amazon EventBridge 事件

您可以使用 EventBridge，透過規則將事件路由傳送至特定的 AWS Lambda 函數。

在本教學課程中，您將使用 Lambda 主控台建立用作 EventBridge 規則目標的函數。然後，您將建立封存和規則，以使用 EventBridge 主控台封存測試事件。一旦封存中有事件，您可重播這些事件。

步驟：

- 步驟 1：建立 Lambda 函數
- 步驟 2：建立封存
- 步驟 3：建立規則
- 步驟 4：傳送測試事件
- 步驟 5：重播事件
- 步驟 6：清理您的資源

步驟 1：建立 Lambda 函數

首先，建立 Lambda 函數以記錄事件。

若要建立 Lambda 函數：

1. 於 AWS Lambdahttps://console.aws.amazon.com/lambda/ 開啟 主控台。
2. 選擇 建立函數。
3. 選擇 Author from scratch (從頭開始撰寫)。
4. 輸入 Lambda 函數的名稱和描述。例如，將函數命名為 `LogScheduledEvent`。
5. 將其餘選項保留為預設值並選擇建立函數。
6. 在函數頁面的程式碼標籤上，按兩下 index.js。
7. 將現有的 JavaScript 程式碼取代為以下程式碼：

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogScheduledEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
```

```
};
```

8. 選擇部署。

步驟 2：建立封存

接下來，建立將保存所有測試事件的封存。

若要建立封存

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇封存。
3. 選擇建立封存。
4. 輸入封存的名稱與描述。例如，命名封存 `ArchiveTest`。
5. 將其餘選項保留為預設值並選擇下一步。
6. 選擇建立封存。

步驟 3：建立規則

建立封存傳送至事件匯流排的規則。

建立規則

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `ARTestRule`。

   在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 預設值。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 中，選擇 Other (其他)。

9. 針對事件模式，請輸入：

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. 選擇 Next (下一步)。
11. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
12. 對於選取目標，請從下拉式清單中選擇 Lambda 函數。
13. 在函數中，選取您在步驟 1：建立 Lambda 函數區段中建立的 Lambda 函數。在此範例中，選取 `LogScheduledEvent`。
14. 選擇 Next (下一步)。
15. 選擇 Next (下一步)。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：傳送測試事件

由於您已設定封存和規則，我們將傳送測試事件以確保封存正常運作。

> **Note**
>
> 事件可能需要一些時間才能得到封存。

若要傳送測試事件 (主控台)

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Event buses (事件匯流排)。
3. 在 預設事件匯流排 圖標中，選擇 動作、傳送事件。
4. 輸入事件來源。例如：`TestEvent`。
5. 對於詳細資訊類型，請輸入 `customerCreated`。
6. 對於事件詳細資訊，請輸入 `{}`。
7. 選擇 Send (傳送)。

步驟 5：重播事件

測試事件在封存中後，您可以對其進行重播。

如要重播封存的事件 (主控台)

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇重播。
3. 選擇重新開始重播。
4. 輸入重播的名稱與描述。例如，命名重播 `ReplayTest`。
5. 在來源中，選取您在步驟 2：建立封存區段中建立的封存。
6. 在重播時間範圍內，請執行下列操作。
   a. 對於開始時間，選取您傳送測試事件的日期以及傳送測試事件之前的時間。例如，`2021/08/11` 和 `08:00:00`。
   b. 對於結束時間，選取目前的日期和時間。例如，`2021/08/11` 和 `09:15:00`。
7. 選擇開始重新播放。

步驟 6：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的函數頁面。
2. 選擇您建立的函數。
3. 選擇 動作 、刪除 。
4. 選擇 Delete (刪除)。

若要刪除 EventBridge 封存

1. 開啟 EventBridge 主控台的封存頁面。
2. 選取您建立的封存。
3. 選擇 Delete (刪除)。
4. 輸入封存名稱，然後選擇刪除。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

建立 Amazon EventBridge 範例應用程式

您可以使用 EventBridge，透過規則將事件路由傳送至特定的 Lambda 函數。

在本教學課程中，您將使用 GitHub儲存庫中的 AWS CLI、Node.js 和程式碼來建立下列項目：

- 產生銀行 ATM 交易事件的 AWS Lambda 函數。
- 用作 EventBridge 規則目標的 Lambda 函數。
- 以及根據事件模式將建立的事件路由傳送至正確下游函數的規則。

此範例使用 AWS SAM 範本來定義 EventBridge 規則。若要進一步瞭解如何搭配 EventBridge 使用 AWS SAM 範本，請參閱 ???。

在儲存庫中，atmProducer 子目錄包含 `handler.js`，其代表產生事件的 ATM 服務。此程式碼是以 Node.js 撰寫的 Lambda 處理常式，並使用這行 JavaScript 程式碼，透過 AWSSDK 將事件發佈至 EventBridge。

```
const result = await eventbridge.putEvents(params).promise()
```

該目錄還包含 `events.js`，列出了一個 Entries 數組中的幾個測試事務。單個事件在 JavaScript 中的定義如下：

```
{
  // Event envelope fields
  Source: 'custom.myATMapp',
  EventBusName: 'default',
  DetailType: 'transaction',
  Time: new Date(),

  // Main event body
  Detail: JSON.stringify({
    action: 'withdrawal',
    location: 'MA-BOS-01',
    amount: 300,
    result: 'approved',
    transactionId: '123456',
    cardPresent: true,
    partnerBank: 'Example Bank',
    remainingFunds: 722.34
  })
```

```
}
```

事件的詳細資訊區段會指定交易屬性。這些措施包括 ATM 的位置、金額、合作銀行以及交易結果。

atmConsumer 子目錄中的 `handler.js` 檔案包含三個函數：

```
exports.case1Handler = async (event) => {
  console.log('--- Approved transactions ---')
  console.log(JSON.stringify(event, null, 2))
}

exports.case2Handler = async (event) => {
  console.log('--- NY location transactions ---')
  console.log(JSON.stringify(event, null, 2))
}

exports.case3Handler = async (event) => {
  console.log('--- Unapproved transactions ---')
  console.log(JSON.stringify(event, null, 2))
}
```

每個函數都會接收交易事件，這些事件會透過 `console.log` 陳述式記錄到 Amazon CloudWatch Logs 中。消費者函數獨立於生產者運行，並且不知道事件來源。

路由邏輯包含在應用程式 AWS SAM 範本所部署的 EventBridge 規則中。這些規則會評估傳入的事件串流，並將相符的事件路由傳送至目標 Lambda 函數。

規則使用事件模式 (屬於 JSON 物件)。JSON 物件的結構與其相符的事件相同。以下是其中一個規則的事件模式。

```
{
  "detail-type": ["transaction"],
  "source": ["custom.myATMapp"],
  "detail": {
    "location": [{
      "prefix": "NY-"
    }]
  }
}
```

步驟：

先決條件

教學課程需要使用以下項目：

- 一個 AWS 帳戶。如果您沒有帳戶，請建立 AWS 帳戶。
- AWS CLI 已安裝。若要安裝 AWS CLI，請參閱安裝、更新和解除安裝 AWS CLI 版本 2。
- 已安裝 Node.js 12.x。若要安裝 Node.js，請參閱下載。

步驟 1：建立應用程式

若要設定範例應用程式，您需要使用 AWS CLI 和 Git 建立所需的 AWS 資源。

建立應用程式

1. 登入 AWS。
2. 安裝 Git 並在本地計算機上安裝 AWS Serverless Application Model CLI。
3. 建立一個新目錄，然後導航到終端中的該目錄。
4. 在命令列輸入 `git clone https://github.com/aws-samples/amazon-eventbridge-producer-consumer-example`：
5. 在命令列中執行以下命令：

```
cd ./amazon-eventbridge-producer-consumer-example
sam deploy --guided
```

6. 在終端中，執行下列動作：

   a. 針對 **`Stack Name`**，請輸入堆疊的名稱。例如，命名堆疊 `Test`。
   b. 針對 **`AWS Region`**，請輸入區域。例如：`us-west-2`。
   c. 針對 **`Confirm changes before deploy`**，請輸入 Y。
   d. 針對 **`Allow SAM CLI IAM role creation`**，請輸入 Y

e. 針對 **Save arguments to configuration file**，請輸入 Y

f. 針對 **SAM configuration file**，請輸入 samconfig.toml。

g. 針對 **SAM configuration environment**，請輸入 default。

步驟 2：執行應用程式

由於您已設定資源，您將使用主控台來測試函數。

執行應用程式

1. 在您部署 AWS SAM 應用程式的相同區域中開啟 Lambda 主控台。
2. 有四個帶有字首 atm-demo 的 Lambda 函數。先選取 atmProducerFn 函數，然後選擇動作，進行測試。
3. 輸入 Test 作為名稱。
4. 選擇 測試 。

步驟 3：檢查日誌並驗證應用程式運作是否正常

由於您已執行應用程式，您將使用主控台來檢查 CloudWatch Logs。

檢查日誌

1. 在您執行 AWS SAM 應用程式的相同區域中開啟 CloudWatch 主控台。
2. 選擇 Logs (日誌)，然後選擇 Log groups (日誌群組)。
3. 選取包含 atmConsumerCase1 的日誌群組。您會看到兩個串流，代表自動櫃員機核准的兩筆交易。選擇日誌串流以檢視輸出。
4. 瀏覽回日誌群組清單，然後選取包含 atmConsumerCase2 的日誌群組。您會看到兩個串流，代表符合紐約地點篩選條件的兩筆交易。
5. 瀏覽回日誌群組清單並選取包含 atmConsumerCase3 的日誌群組。打開串流以查看被拒絕的交易。

步驟 4：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 開啟 EventBridge 主控台的規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 函數頁面。
2. 選擇您建立的函數。
3. 選擇動作、刪除。
4. 選擇 Delete (刪除)。

若要刪除 CloudWatch Logs 日誌群組

1. 開啟 CloudWatch 主控台。
2. 選擇日誌、日誌群組
3. 選取在教學課程中建立的日誌群組。
4. 選擇動作、刪除日誌群組。
5. 選擇 Delete (刪除)。

教學課程：使用 EventBridge 結構描述登錄檔下載事件的程式碼繫結

您可以為事件結構描述產生程式碼繫結，以加速 Golang、Java、Python 和 TypeScript 的開發速度。您可以取得現有 AWS 服務、您建立的結構描述，以及根據事件匯流排上的事件產生結構描述的程式碼繫結。您可以使用下列其中一個結構描述名稱來產生結構描述的程式碼繫結：

- EventBridge 主控台
- EventBridge 結構描述登錄檔 API
- 您的 IDE 與一個 AWS 工具組

在本教學課程中，您將從 AWS 服務事件的 EventBridge 結構描述產生並下載程式碼繫結。

從 EventBridge 結構描述產生程式碼繫結

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Schemas (結構描述)。
3. 選取 AWS 事件結構描述登錄檔 標籤。
4. 透過瀏覽結構描述登錄檔或搜尋結構描述，尋找您要程式碼繫結的 AWS 服務結構描述。
5. 選取結構描述名稱。
6. 在結構描述詳細資訊頁面的版本區段中，選取下載程式碼繫結。
7. 在 Download code bindings (下載程式碼繫結) 頁面上，選取您要下載的程式碼繫結語言。
8. 選取 download (下載)。

   開始下載可能需要幾秒鐘的時間。下載的檔案將是您所選語言的程式碼繫結 .zip 檔案。
9. 解壓縮下載的檔案並將其新增至您的項目中。

   下載的套件包含 README 檔案，說明如何在各種框架中設定套件的相依性。

在您自己的程式碼中使用這些程式碼繫結，以協助快速建置使用此 EventBridge 事件的應用程式。

教學課程：使用輸入轉換器以自訂 EventBridge 傳送至事件目標的內容

在將事件傳送至規則目標之前，您可以使用 EventBridge 中的輸入轉換器自訂事件中的文字。

為此，您可以定義事件的 JSON 路徑，並為其輸出指派不同的變數。然後，您可以在輸入範本中使用這些變數。字元 < 和 > 不可逸出。如需詳細資訊，請參閱 Amazon EventBridge 輸入轉換

> **Note**
>
> 如果您指定變數對應不存在於事件中的 JSON 路徑，則該變數不會建立，且不會出現在輸出中。

在該教學課程中，您將建立事件與 `detail-type: "customerCreated"` 相符的規則。輸入轉換器會將 `type` 變數對應至事件中的 $.detail-type JSON 路徑。然後，EventBridge 會將變數放入輸入範本「這個事件是 <類型>」。結果是下列 Amazon SNS 訊息。

```
"This event was of customerCreated type."
```

步驟：

- 步驟 1：建立 Amazon SNS 主題
- 步驟 2：建立 Amazon SNS 訂閱
- 步驟 3：建立規則
- 步驟 4：傳送測試事件
- 步驟 5：確認成功
- 步驟 6：清理您的資源

步驟 1：建立 Amazon SNS 主題

建立主題以接收來自 EventBridge 的事件。

若要建立主題

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Topics (主題)。
3. 請選擇建立主題。

4. 針對 Type (類型)，選擇 Standard (標準)。
5. 輸入 **`eventbridge-IT-test`** 作為主題的名稱。
6. 請選擇建立主題。

步驟 2：建立 Amazon SNS 訂閱

建立訂閱以取得含有轉換後的資訊的電子郵件。

若要建立訂閱

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Subscriptions (訂閱)。
3. 選擇建立訂閱。
4. 在主題 ARN 中，選擇您在步驟 1 建立的主題。在本教學課程中，選擇 eventbridge-IT-test。
5. 對於 Protocol (通訊協定)，選擇 Email (電子郵件)。
6. 針對 Endpoint (端點)，輸入電子郵件地址。
7. 選擇建立訂閱。
8. 在您從 AWS 通知收到的電子郵件中選擇確認訂閱，以確認訂閱。

步驟 3：建立規則

建立規則以使用輸入轉換器自訂前往目標的執行個體狀態資訊。

建立規則

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `ARTestRule`
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取預設值。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。

8. 在 Event source (事件來源) 中，選擇 Other (其他)。
9. 針對事件模式，請執行下列動作：

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. 選擇 Next (下一步)。
11. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
12. 針對選取目標，從下拉式清單中選擇 SNS 主題。
13. 對於主題，選取您在步驟 1 建立的 Amazon SNS 主題。在本教學課程中，選擇 eventbridge-IT-test。
14. 對於其他設定，執行下列動作：
    a. 對於設定目標輸入，請從下拉式清單中選擇輸入轉換器。
    b. 選擇設定輸入轉換器。
    c. 針對範例事件，請輸入下列內容：

```
{
  "detail-type": "customerCreated"
}
```

    d. 對於目標輸入轉換器，請執行下列動作：
        i. 對於輸入路徑，請輸入下列內容：

```
{"detail-type":"$.detail-type"}
```

        ii. 對於輸入範本，請輸入下列內容：

```
"This event was of <detail-type> type."
```

    e. 選擇 確認。
15. 選擇 Next (下一步)。
16. 選擇 Next (下一步)。
17. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：傳送測試事件

現在您已設定 SNS 主題和規則，我們將傳送測試事件以確保規則正常運作。

若要傳送測試事件 (主控台)

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Event buses (事件匯流排)。
3. 在 預設事件匯流排 圖標中，選擇 動作，傳送事件。
4. 輸入事件來源。例如：`TestEvent`。
5. 對於詳細資訊類型，請輸入 `customerCreated`。
6. 對於事件詳細資訊，請輸入 `{}`。
7. 選擇 Send (傳送)。

步驟 5：確認成功

如果您收到符合預期輸出的 AWS 通知的電子郵件，表示您已成功完成教學課程。

步驟 6：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 SNS 主題

1. 開啟 SNS 主控台的主題頁面。
2. 選擇您建立的主題。
3. 選擇 Delete (刪除)。
4. 輸入 **`delete me`**。
5. 選擇 Delete (刪除)。

刪除 SNS 訂閱

1. 在 SNS 主控台開啟訂閱頁面。
2. 選取您建立的訂閱。
3. 選擇 Delete (刪除)。

4. 選擇 Delete (刪除)。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

與其他 AWS 服務整合的 Amazon EventBridge 教學課程

Amazon EventBridge 可與其他 AWS 服務搭配使用，以處理 事件 或叫用 AWS 資源作為 規則 的 目標。以下教學課程說明如何將 EventBridge 與其他 AWS 服務整合。

教學：

- 教學課程：使用 EventBridge 記錄 Auto Scaling 群組的狀態
- 教學課程：使用 EventBridge 記錄 AWS API 呼叫
- 教學課程：使用 EventBridge 來記錄 Amazon EC2 執行個體的狀態
- 教學課程：使用 EventBridge 記錄 Amazon S3 物件層級操作
- 教學課程：使用 EventBridge 和 aws.events 結構描述將事件傳送至 Amazon Kinesis 串流
- 教學課程：使用 EventBridge 排定自動化 Amazon EBS 快照
- 教學課程：建立 Amazon S3 物件時傳送通知。
- 教學課程：使用 EventBridge 排定 AWS Lambda 函數

教學課程：使用 EventBridge 記錄 Auto Scaling 群組的狀態

您可以執行 AWS Lambda 函數以在 Auto Scaling 群組啟動或終止 Amazon EC2 執行個體 (指示事件是否成功) 時記錄 事件。

如需關於使用 Amazon EC2 Auto Scaling 事件的更多案例的資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》 中的 使用 EventBridge 來處理 Auto Scaling 事件。

在本教學課程中，您會建立 Lambda 函數並在 EventBridge 主控台中建立 規則，該規則會在 Amazon EC2 Auto Scaling 群組啟動或終止執行個體時調用該函數。

步驟：

- 先決條件
- 步驟 1：建立 Lambda 函數
- 步驟 2：建立規則
- 步驟 3：測試規則
- 步驟 4：確認成功
- 步驟 5：清理您的資源

先決條件

教學課程需要使用以下項目：

- Auto Scaling 群組。如需關於建立群組的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 使用啟動組態建立 Auto Scaling 群組。

步驟 1：建立 Lambda 函數

建立 Lambda 函數以記錄 Auto Scaling 群組的擴展和縮小事件。

建立 Lambda 函數

1. 在 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 建立函數。
3. 選擇 Author from scratch (從頭開始撰寫)。
4. 輸入 Lambda 函數的名稱。例如，將函數命名為 `LogAutoScalingEvent`。
5. 將其餘選項保留為預設值並選擇 建立函數。

6. 在函數頁面的 程式碼 標籤上，按兩下 index.js。
7. 將現有的程式碼取代為以下程式碼。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogAutoScalingEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. 選擇 Deploy (部署)。

步驟 2：建立規則

建立規則來執行您在步驟 1 中建立的 Lambda 函數。當您的 Auto Scaling 群組啟動或停止執行個體時，規則便會執行。

建立規則

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `TestRule`
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 預設值。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 欄位中，選擇 AWS services (服務)。
9. 針對 Event pattern (事件模式) 請執行下列動作：
    a. 對於 事件來源，請從下拉式清單中選取 Auto Scaling。
    b. 對於 事件類型，請從下拉式清單中選取 執行個體啟動和終止。
    c. 選擇 任何執行個體事件 和 任何群組名稱。
10. 選擇 Next (下一步)。

11. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
12. 對於 選取目標，請從下拉式清單中選擇 Lambda 函數。
13. 在 函數 中，選取您在 步驟 1：建立 Lambda 函數 區段中建立的 Lambda 函數。在此範例中，選取 `LogAutoScalingEvent`。
14. 選擇 Next (下一步)。
15. 選擇 Next (下一步)。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 3：測試規則

您可以手動擴展 Auto Scaling 群組以啟動執行個體，藉此測試您的規則。等待幾分鐘讓擴展事件發生，然後驗證您的 Lambda 函數是否被叫用。

使用 Auto Scaling 群組測試您的規則

1. 若要增加 Auto Scaling 群組的大小，請執行下列動作：
   a. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
   b. 在導覽窗格中，選擇 Auto Scaling (Auto Scaling)、Auto Scaling Groups (Auto Scaling 群組)。
   c. 選取您的 Auto Scaling 群組的核取方塊。
   d. 在 Details (詳細資訊) 標籤上，選擇 Edit (編輯)。在 Desired (所需) 中提高所需的容量 1。例如，如果目前值為 2，則輸入 3。所需容量必須小於或等於該群組的大小上限。如果您的 Desired (所需) 的新數值大於 Max (最大值)，則您必須更新 Max (最大值) 設定。完成後，請選擇 Save (儲存)。
2. 若要檢視 Lambda 函數的輸出，請執行下列動作：
   a. 前往 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
   b. 在導覽窗格中，選擇 Logs (日誌)。
   c. 為 Lambda 函數 (`/aws/lambda/`*`function-name`*) 選取日誌群組名稱。
   d. 選取日誌串流的名稱以檢視函數為您啟動的執行個體所提供的資料。
3. (選用) 完成後，您可以將所需的容量減少 1，如此一來，Auto Scaling 群組將恢復為之前的大小。

步驟 4：確認成功

如果您在 CloudWatch logs 中看到 Lambda 事件，表示您已成功完成本教學課程。如果事件不在 CloudWatch logs中，請驗證規則是否已成功建立，開始進行故障診斷，如果規則看起來正確，請驗證 Lambda 函數的程式碼是否正確無誤。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟 規則頁面。
2. 選取您建立的規則。
3. 選擇 刪除。
4. 選擇 刪除。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 函數頁面。
2. 選擇您建立的函數。
3. 選擇 動作、刪除。
4. 選擇 刪除。

教學課程：使用 EventBridge 記錄 AWS API 呼叫

您可以使用 Amazon EventBridge 規則來回應由 AWS CloudTrail 記錄的 AWS 服務所發出的 API 呼叫。

在本教學課程中，您會在 EventBridge 主控台中建立 AWS CloudTrail 追蹤、Lambda 函數和規則。當 Amazon EC2 執行個體停止時，規則會叫用 Lambda 函數。

步驟：

- 步驟 1：建立 AWS CloudTrail 追蹤
- 步驟 2：建立 AWS Lambda 函數
- 步驟 3：建立規則
- 步驟 4：測試規則
- 步驟 5：確認成功
- 步驟 6：清理您的資源

步驟 1：建立 AWS CloudTrail 追蹤

如果您已設定追蹤，請跳到步驟 2。

若要建立追蹤記錄

1. 前往 https://console.aws.amazon.com/cloudtrail/ 開啟 CloudTrail 主控台。
2. 選擇 Trails (追蹤)、Create trail (建立追蹤)。
3. 在 Trail name (追蹤名稱) 中輸入追蹤的名稱。
4. 在 儲存位置 的 建立新的 S3 儲存貯體。
5. 對於 AWS KMS 別名，請輸入 KMS 金鑰的別名。
6. 選擇 Next (下一步)。
7. 選擇 Next (下一步)。
8. 選擇 Create trail (建立追蹤)。

步驟 2：建立 AWS Lambda 函數

建立 Lambda 函數以記錄 API 呼叫事件。

建立 Lambda 函數

1. 前往 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 建立函數。
3. 選擇 Author from scratch (從頭開始撰寫)。
4. 輸入 Lambda 函數的名稱和描述。例如，將函數命名為 `LogEC2StopInstance`。
5. 將其餘選項保留為預設值並選擇建立函數。
6. 在函數頁面的程式碼標籤上，按兩下 index.js。
7. 將現有的程式碼取代為以下程式碼。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2StopInstance');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. 選擇 部署 。

步驟 3：建立規則

建立規則，在您停止 Amazon EC2 執行個體時，執行您在步驟 2 建立的 Lambda 函數。

建立規則

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `TestRule`
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 預設值。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 欄位中，選擇 AWS services (服務)。

9. 針對 Event pattern (事件模式) 請執行下列動作：
    a. 對於事件來源，請從下拉式清單中選取 EC2。
    b. 對於事件類型，請從下拉式清單中選取 AWS 透過 CloudTrail 進行的 API 呼叫。
    c. 選擇特定動作 並輸入 `StopInstances`。
10. 選擇 Next (下一步)。
11. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
12. 對於選取目標，請從下拉式清單中選擇 Lambda 函數。
13. 在函數中，選取您在 步驟 1：建立 Lambda 函數區段中建立的 Lambda 函數。在此範例中，選取 `LogEC2StopInstance`。
14. 選擇 Next (下一步)。
15. 選擇 Next (下一步)。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：測試規則

您可以使用 Amazon EC2 主控台停用 Amazon EC2 執行個體，以測試您的規則。等待幾分鐘讓執行個體停止，然後在 CloudWatch 主控台檢查 AWS Lambda 指標以確認您的函數已被執行。

停用執行個體以測試您的規則

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 啟動執行個體。如需詳細資訊，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的 啟動您的執行個體。
3. 停止執行個體。如需詳細資訊，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的停止和啟動執行個體。
4. 若要檢視 Lambda 函數的輸出，請執行下列動作：
    a. 前往 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
    b. 在導覽窗格中，選擇 Logs (日誌)。
    c. 為 Lambda 函數 (`/aws/lambda/`*`function-name`*) 選取日誌群組名稱。
    d. 選取日誌串流的名稱以檢視函數為您停止的執行個體提供的資料。
5. (選用) 完成後，請終止已停止的執行個體。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的終止您的執行個體。

步驟 5：確認成功

如果您在 CloudWatch logs 中看到 Lambda 事件，表示您已成功完成本教學課程。如果事件不在 CloudWatch logs 中，請驗證規則是否已成功建立，開始進行故障診斷，如果規則看起來正確，請驗證 Lambda 函數的程式碼是否正確無誤。

步驟 6：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟 規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 函數頁面。
2. 選擇您建立的函數。
3. 選擇動作、刪除。
4. 選擇 Delete (刪除)。

若要刪除 CloudTrail 線索。

1. 開啟 CloudTrail 主控台的 Trails (線索) 頁面。
2. 選擇您建立的線索。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

教學課程：使用 EventBridge 來記錄 Amazon EC2 執行個體的狀態

您可以建立一個 AWS Lambda 函數，記錄 Amazon EC2 執行個體狀態的變化。您可以建立規則，當有狀態轉換或有轉移到一或多個有興趣的狀態時執行 Lambda 函數。在此教學中，您將記錄任何新執行個體的啟動。

步驟：

- 步驟 1：建立 AWS Lambda 函數
- 步驟 2：建立規則
- 步驟 3：測試規則
- 步驟 4：確認成功
- 步驟 5：清理您的資源

步驟 1：建立 AWS Lambda 函數

建立 Lambda 函數以記錄狀態變更事件。當您在步驟 2 建立規則時指定此函數。

建立 Lambda 函數

1. 在 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 建立函數。
3. 選擇 Author from scratch (從頭開始撰寫)。
4. 輸入 Lambda 函數的名稱和描述。例如，將函數命名為 `LogEC2InstanceStateChange`。
5. 將其餘選項保留為預設值並選擇建立函數。
6. 在函數頁面的程式碼標籤上，按兩下 index.js。
7. 將現有的程式碼取代為以下程式碼。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2InstanceStateChange');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. 選擇 Deploy (部署)。

步驟 2：建立規則

建立規則，執行您在步驟 1 中建立的 Lambda 函數。規則會在您啟動 Amazon EC2 執行個體時執行。

若要建立 EventBridge 規則

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `TestRule`
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 預設值。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 欄位中，選擇 AWS services (服務)。
9. 針對 Event pattern (事件模式) 請執行下列動作：
    a. 對於事件來源，從下拉式清單中選取 EC2。
    b. 在事件類型中，從下拉式清單中選擇 EC2 執行個體狀態變更通知。
    c. 選擇特定狀態並從下拉式清單中選擇正在執行。
    d. 選擇任何執行個體。
10. 選擇 Next (下一步)。
11. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
12. 對於選取目標，請從下拉式清單中選擇 Lambda 函數。
13. 在函數中，選取您在步驟 1：建立 Lambda 函數 區段中建立的 Lambda 函數。在此範例中，選取 `LogEC2InstanceStateChange`。
14. 選擇 Next (下一步)。
15. 選擇 Next (下一步)。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 3：測試規則

您可以使用 Amazon EC2 主控台停用 Amazon EC2 執行個體，以測試您的規則。等待幾分鐘讓執行個體停止，然後在 CloudWatch 主控台檢查 AWS Lambda 指標以確認您的函數已被執行。

停用執行個體以測試您的規則

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 啟動執行個體。如需詳細資訊，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的啟動您的執行個體。
3. 停止執行個體。如需詳細資訊，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的停止和啟動執行個體。
4. 若要檢視 Lambda 函數的輸出，請執行下列動作：
   a. 前往 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
   b. 在導覽窗格中，選擇 Logs (日誌)。
   c. 為 Lambda 函數 (`/aws/lambda/function-name`) 選取日誌群組名稱。
   d. 選取日誌串流的名稱以檢視函數為您停止的執行個體提供的資料。
5. (選用) 完成後，請終止已停止的執行個體。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的終止您的執行個體。

步驟 4：確認成功

如果您在 CloudWatch logs 中看到 Lambda 事件，表示您已成功完成本教學課程。如果事件不在 CloudWatch logs中，請驗證規則是否已成功建立，開始進行故障診斷，如果規則看起來正確，請驗證 Lambda 函數的程式碼是否正確無誤。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟規則頁面。
2. 選取您建立的規則。
3. 選擇刪除。

4. 選擇刪除。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 函數頁面。
2. 選擇您建立的函數。
3. 選擇動作、刪除。
4. 選擇刪除。

教學課程：使用 EventBridge 記錄 Amazon S3 物件層級操作

您可以在您的 Amazon S3 儲存貯體記錄物件層級 API 操作。您必須使用 AWS CloudTrail 設定追蹤以接收這些事件，Amazon EventBridge 才能符合這些事件。

在本教學課程中，您會建立 CloudTrail 追蹤，建立 AWS Lambda 函數，然後在 EventBridge 主控台中建立規則，以叫用該函數以回應 S3 資料事件。

步驟：

- 步驟 1：設定您的 AWS CloudTrail 追蹤
- 步驟 2：建立 AWS Lambda 函數
- 步驟 3：建立規則
- 步驟 4：測試 規則
- 步驟 5：確認成功
- 步驟 6：清理您的資源

步驟 1：設定您的 AWS CloudTrail 追蹤

若要將 S3 儲存貯體資料事件記錄至 AWS CloudTrail 和 EventBridge，請首先建立追蹤。追蹤將擷取您帳戶中的 API 呼叫和相關事件，然後將日誌檔案傳送到您指定的 S3 儲存貯體。您可以更新現有的追蹤或建立新的追蹤。

如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的資料事件。

若要建立追蹤記錄

1. 透過 https://console.aws.amazon.com/cloudtrail/ 開啟 CloudTrail 主控台。
2. 選擇 Trails (追蹤)、Create trail (建立追蹤)。
3. 在 Trail name (追蹤名稱) 中輸入追蹤的名稱。
4. 在儲存位置的建立新的 S3 儲存貯體中。
5. 針對 AWS KMS 別名，輸入 KMS 金鑰的別名。
6. 選擇下一步。
7. 針對事件類型，選擇資料事件
8. 針對資料事件，執行下列其中一項操作：

- 若要記錄儲存貯體中所有 Amazon S3 物件的資料事件，請指定 S3 儲存貯體和空的前綴。當事件發生在該儲存貯體中的物件上時，追蹤即會處理並記錄此事件。
- 若要記錄儲存貯體中特定 Amazon S3 物件的資料事件，請指定 S3 儲存貯體和物件字首。當事件發生在該儲存貯體的物件上，而此物件是以指定的前綴開頭，線索就會處理並記錄此事件。

9. 對於每個資源，選擇是否要記錄讀取事件、寫入事件或兩者。
10. 選擇下一步。
11. 選擇 Create trail (建立追蹤)。

步驟 2：建立 AWS Lambda 函數

建立 Lambda 函數以記錄 S3 儲存貯體的資料事件。

建立 Lambda 函數

1. 在 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 建立函數。
3. 選擇 Author from scratch (從頭開始撰寫)。
4. 輸入 Lambda 函數的名稱和描述。例如，將函數命名為 `LogS3DataEvents`。
5. 將其餘選項保留為預設值並選擇建立函數。
6. 在函數頁面的程式碼標籤上，按兩下 index.js。
7. 將現有的程式碼取代為以下程式碼。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogS3DataEvents');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. 選擇 部署 。

步驟 3：建立規則

建立規則來執行您在步驟 2 中建立的 Lambda 函數。執行此規則旨在回應 Amazon S3 資料事件。

建立規則

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `TestRule`
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取預設值。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇下一步。
8. 在 Event source (事件來源) 欄位中，選擇 AWS services (服務)。
9. 針對 Event pattern (事件模式) 請執行下列動作：
   a. 在事件來源下，從下拉式清單中，選擇 Simple Storage Service (S3)。
   b. 對於事件類型，請從下拉式清單中選取 透過 CloudTrail 進行的物件層級 API 呼叫。
   c. 選擇 Specific operation(s) (特定操作)，然後選擇 PutObject (PutObject)。
   d. 在預設情況下，規則符合區域中所有儲存貯體的資料事件。為了符合特定儲存貯體的資料事件，選擇 Specify bucket(s) by name (以名稱指定儲存貯體)，然後輸入一或多個儲存貯體。
10. 選擇下一步。
11. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
12. 對於選取目標，請從下拉式清單中選擇 Lambda 函數。
13. 針對函數，選取您在步驟 1 建立的 `LogS3DataEvents` Lambda 函數。
14. 選擇下一步。
15. 選擇下一步。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：測試 規則

若要測試規則，請將物件放入您的 S3 儲存貯體。您可以驗證您的 Lambda 函數是否被叫用。

檢視 Lambda 函數的日誌

1. 前往 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。

2. 在導覽窗格中，選擇 Logs (日誌)。
3. 為 Lambda 函數 (/aws/lambda/*function-name*) 選取日誌群組名稱。
4. 選取日誌串流的名稱以檢視函數為您啟動的執行個體所提供的資料。

您也可以在為您的追蹤所指定的 S3 儲存貯體中檢查 CloudTrail logs。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的取得和檢視 CloudTrail 日誌檔案。

步驟 5：確認成功

如果您在 CloudWatch logs 中看到 Lambda 事件，表示您已成功完成本教學課程。如果事件不在 CloudWatch logs 中，請驗證規則是否已成功建立，開始進行故障診斷，如果規則看起來正確，請驗證 Lambda 函數的程式碼是否正確無誤。

步驟 6：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 函數頁面。
2. 選擇您建立的函數。
3. 選擇動作、刪除。
4. 選擇 Delete (刪除)。

若要刪除 CloudTrail 追蹤

1. 開啟 CloudTrail 主控台的 Trails (線索) 頁面。
2. 選擇您建立的追蹤。

3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

教學課程：使用 EventBridge 和 **aws.events** 結構描述將事件傳送至 Amazon Kinesis 串流

您可以將 EventBridge 中的 AWS API 呼叫事件傳送至 Amazon Kinesis 串流、建立 Kinesis Data Streams 應用程式，以及處理大量資料。在本教學課程中，您會建立 Kinesis 串流，然後在 EventBridge 主控台中建立 規則。該規則會在 Amazon EC2 執行個體停止時將事件傳送至該串流。

步驟：

先決條件

在本教學課程中，您將使用下列內容：

- 使用 AWS CLI 與 Kinesis 串流搭配使用。

  若要安裝 AWS CLI，請參閱安裝、更新和解除安裝 AWS CLI 版本 2。

> **Note**
>
> 本教學課程使用 AWS 事件和內置的 `aws.events` 結構描述登錄檔。您也可以手動將自訂結構描述新增至自訂結構描述登錄檔，或使用結構描述探索，以根據自訂事件的結構描述建立 EventBridge 規則。
> 如需結構描述的詳細資訊，請參閱 ???。如需關於使用其他事件模式選項建立規則的詳細資訊，請參閱 ???。

步驟 1：建立 Amazon Kinesis 串流

若要建立串流，請在命令提示中使用 `create-stream` AWS CLI 命令。

```
aws kinesis create-stream --stream-name test --shard-count 1
```

當串流狀態為 ACTIVE 時，串流已備妥。若要檢查串流狀態，請使用 describe-stream 命令。

```
aws kinesis describe-stream --stream-name test
```

步驟 2：建立規則

建立一個規則，在您停止 Amazon EC2 執行個體時，將事件傳送至您的串流。

建立 規則

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 TestRule
5. 針對事件匯流排，選取預設值。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。
9. 在建立方法中，選擇使用結構描述。
10. 針對 Event pattern (事件模式) 請執行下列動作：

    a. 針對結構描述類型，從結構描述登記檔選擇結構描述。

    b. 針對結構描述登錄檔，請從下拉式清單中選擇 aws.events

    c. 在結構描述中，從下拉式清單中選擇 aws.ec2@EC2InstanceStateChangeNotification。

    EventBridge 會在模型下顯示事件結構描述。

    EventBridge 會在事件而非事件模式所需的任何屬性旁邊顯示紅色星號。

    d. 在模型中，設定下列事件篩選屬性：

    i. 選取狀態屬性旁邊的 +編輯。

    將關係留空。於 Value (數值) 輸入 running。選擇設定。

ii. 選取來源屬性旁邊的+ 編輯。

將關係留空。於 Value (數值) 輸入 `aws.ec2`。選擇設定。

iii. 選取詳細資料類型屬性旁邊的+ 編輯。

將關係留空。於 Value (數值) 輸入 `EC2 Instance State-change Notification`。選擇設定。

e. 若要檢視您已建構的事件模式，請選擇在 JSON 中產生事件模式

EventBridge 會以 JSON 格式顯示事件模式：

```
{
  "detail": {
    "state": ["running"]
  },
  "detail-type": ["EC2 Instance State-change Notification"],
  "source": ["aws.ec2"]
}
```

11. 選擇 Next (下一步)。
12. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
13. 針對選取目標，從下拉式清單中選擇 Kinesis 串流。
14. 在串流中，選取您在步驟 1：建立 Amazon Kinesis 串流 區段中建立的 Kinesis 串流。在此範例中，選取 `test`。
15. 針對執行角色，請選擇為此特定資源建立新角色。
16. 選擇 Next (下一步)。
17. 選擇 Next (下一步)。
18. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 3：測試規則

為了測試您的規則，請停止 Amazon EC2 執行個體。等待幾分鐘讓執行個體停止，然後檢查 CloudWatch 指標以確認您的函數是否已被執行。

停用執行個體以測試您的規則

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

2. 啟動執行個體。如需詳細資訊，請參閱《適用於 Linux 執行個體的 Amazon EC2 使用者指南》中的啟動您的執行個體。
3. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
4. 在導覽窗格中，選擇 Rules (規則)。

   選擇您建立的規則名稱，並選擇 Metrics for the rule (規則的指標)。
5. (選用) 完成後，請終止執行個體。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的終止您的執行個體。

步驟 4：確認事件是否已傳送

您可以使用 AWS CLI 從串流取得記錄以確認事件已傳送。

取得記錄

1. 若要開始從 Kinesis 串流讀取，請在命令提示中使用 `get-shard-iterator` 命令。

```
aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterator-
type TRIM_HORIZON --stream-name test
```

   下列為範例輸出。

```
{
    "ShardIterator": "AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg="
}
```

2. 若要取得記錄，請使用以下 `get-records` 命令。根據上一步驟的輸出使用分片反覆運算器。

```
aws kinesis get-records --shard-
iterator AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg=
```

   如果命令成功，它會請求來自您的串流的指定碎片的記錄。您可能會收到零或多筆記錄。傳回的任何記錄可能不會呈現您串流中的所有記錄。如果您沒有收到預期的資料，請呼叫 `get-records`。

3. Kinesis 中的記錄用 Base64 編碼。使用 Base64 解碼器對資料進行解碼，以便您可以驗證這是以 JSON 格式傳送至串流的事件。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟 規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

刪除 Kinesis 串流

1. 開啟 Kinesis 主控台的資料串流頁面。
2. 選擇您建立的串流。
3. 選擇動作、刪除。
4. 在欄位中輸入刪除 並選擇 刪除。

教學課程：使用 EventBridge 排定自動化 Amazon EBS 快照

您可以按照排程執行 EventBridge 規則。在本教學課程中，您會依排程建立現有 Amazon Elastic Block Store (Amazon EBS) 磁碟區的快照。您可以選擇固定速率以每隔幾分鐘建立快照，或使用 cron 運算式在特定時間建立快照。

> **Important**
>
> 若要使用內建目標建立規則，您必須使用 AWS Management Console。

步驟：

- 步驟 1：建立規則
- 步驟 2：測試規則
- 步驟 3：確認成功
- 步驟 4：清理您的資源

步驟 1：建立規則

建立排程拍攝快照的規則。您可以使用 rate 表達式或 cron 表達式來指定排程。如需更多詳細資訊，請參閱 建立依排程執行的 Amazon EventBridge 規則。。

建立規則

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。

   在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 AWS default event bus (AWS 預設事件匯流排)。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型)，選擇 Schedule (排程)。
7. 選擇 Next (下一步)。

8. 對於 排程模式，請選擇 以一般儲存費率執行的排程，例如每 10 分鐘執行一次。 輸入 **5** 並從下拉式清單中選擇 分鐘。
9. 選擇 Next (下一步)。
10. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
11. 針對選取目標，從下拉式清單中選擇 EBS 建立快照。
12. 針對磁碟區 ID，輸入 Amazon EBS 磁碟區的磁碟區 ID。
13. 針對執行角色，選擇為此特定資源建立新角色。
14. 選擇 Next (下一步)。
15. 選擇 Next (下一步)。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 2：測試規則

您可以在建立第一個快照之後，藉由檢視該快照以驗證您的規則。

測試規則

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在導覽窗格中，選擇 Elastic Block Store、Snapshots (快照)。
3. 確定第一個快照出現在清單中。

步驟 3：確認成功

如果您在清單中看到快照，表示您已成功完成本教學課程。如果快照不在清單中，請驗證規則已成功建立以開始故障診斷。

步驟 4：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟 規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。

4. 選擇 Delete (刪除)。

教學課程：建立 Amazon S3 物件時傳送通知。

當使用 Amazon EventBridge 和 Amazon SNS 建立 Amazon Simple Storage Service (Amazon S3) 物件時，您可以傳送電子郵件通知。在本教學課程中，您將建立 SNS 主題和訂閱。然後，您將在 EventBridge 主控台中建立規則，該規則會在收到 Amazon S3 `Object Created` 事件時將事件傳送至該主題。

步驟：

先決條件

若要在 EventBridge 接收 Amazon S3 事件，您必須在 Amazon S3 主控台中啟用 EventBridge。本教學課程假設 EventBridge 已啟用。如需詳細資訊，請參閱 S3 主控台中啟用 Amazon EventBridge。

步驟 1：建立 Amazon SNS 主題

建立主題以接收來自 EventBridge 的事件。

若要建立主題

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Topics (主題)。
3. 請選擇 建立主題。
4. 針對 Type (類型)，選擇 Standard (標準)。
5. 輸入 **`eventbridge-test`**，作為主題的名稱。
6. 請選擇 建立主題。

步驟 2：建立 Amazon SNS 訂閱

建立訂閱以在主題收到事件時從 Amazon S3 取得電子郵件通知。

若要建立訂閱

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Subscriptions (訂閱)。
3. 選擇建立訂閱。
4. 在主題 ARN 中，選擇您在步驟 1 建立的主題 在本教學課程中，選擇 eventbridge-test。
5. 對於 Protocol (通訊協定)，選擇 Email (電子郵件)。
6. 針對 Endpoint (端點)，輸入電子郵件地址。
7. 選擇建立訂閱。
8. 透過從 AWS 通知收到的電子郵件中選擇確認訂閱，確認訂閱。

步驟 3：建立規則

建立 Amazon S3 物件時，建立將事件傳送至您的主題的規則。

建立規則

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，命名規則 `s3-test`
5. 針對事件匯流排，選取預設值。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。
7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 欄位中，選擇 AWS events or EventBridge partner events (事件或 EventBridge 合作夥伴事件)。
9. 針對建立方法，選取使用模式表單。
10. 針對 Event pattern (事件模式) 請執行下列動作：
    a. 針對事件來源，請從下拉式清單中選取 AWS 服務。
    b. 針對AWS 服務，請從下拉式清單中選取 Simple Storage Service (S3)。
    c. 對於事件類型，請從下拉式清單中選擇 Amazon S3 事件通知。
    d. 選擇特定事件並從下拉式清單中選擇建立物件。

e. 選擇任何儲存貯體。

11. 選擇 Next (下一步)。
12. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
13. 對於選取目標，從下拉式清單中選取 SNS 主題。
14. 在主題中，選取您在步驟 1：建立 SNS 主題區段中建立的 Amazon SNS 主題。在此範例中，選取 `eventbridge-test`。
15. 選擇 Next (下一步)。
16. 選擇 Next (下一步)。
17. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：測試規則

若要測試您的規則，請將檔案上傳到啟用 EventBridge 的儲存貯體，以建立 Amazon S3 物件。然後，請稍待幾分鐘並確認您是否收到來自 AWS 通知的電子郵件。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 SNS 主題

1. 開啟 SNS 主控台的主題頁面。
2. 選擇您建立的主題。
3. 選擇 Delete (刪除)。
4. 輸入 **`delete me`**。
5. 選擇 Delete (刪除)。

刪除 SNS 訂閱

1. 在 SNS 主控台開啟訂閱頁面。
2. 選取您建立的訂閱。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

教學課程：使用 EventBridge 排定 AWS Lambda 函數

您可以設定規則來排程執行 AWS Lambda 函數。本教學說明如何使用 AWS Management Console 或 AWS CLI 建立規則。如果您想要使用 AWS CLI 但尚未安裝，請參閱安裝、更新和解除安裝 AWS CLI 版本2。

對於排程，EventBridge 在排程表達式中不提供第二層級的精確度。使用 cron 表達式的最小解析是一分鐘。由於 EventBridge 和目標服務的分散式特性，觸發排程規則的時間與目標服務執行目標資源的時間之間的延遲可能是幾秒鐘。

步驟：

- 步驟 1：建立 Lambda 函數
- 步驟 2：建立規則
- 步驟 3：驗證規則
- 步驟 4：確認成功
- 步驟 5：清理您的資源

步驟 1：建立 Lambda 函數

建立 Lambda 函數以記錄排程事件。

建立 Lambda 函數

1. 在 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 建立函數。
3. 選擇 Author from scratch (從頭開始撰寫)。
4. 輸入 Lambda 函數的名稱和描述。例如，將函數命名為 `LogScheduledEvent`。
5. 將其餘選項保留為預設值並選擇建立函數。
6. 在函數頁面的程式碼標籤上，按兩下 index.js。
7. 將現有的程式碼取代為以下程式碼。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogScheduledEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
```

```
    callback(null, 'Finished');
};
```

8. 選擇 Deploy (部署)。

步驟 2：建立規則

建立規則以依排程執行步驟 1 中建立的 Lambda 函數。

您可以使用主控台或 AWS CLI 建立規則。如要使用 AWS CLI，您必須先授予規則許可以叫用 Lambda 函數。然後，您可以建立規則，並新增 Lambda 函數做為目標。

建立規則 (主控台)

1. 在 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。

   在同一個區域和同一個事件匯流排上，規則不能與另一個規則同名。
5. 針對 Event bus (事件匯流排)，選擇要與此規則建立關聯的事件匯流排。如果您想要此規則匹配來自您的帳戶的事件，請選取 AWS default event bus (AWS 預設事件匯流排)。當您帳戶中的 AWS 服務發出事件時，一律會前往您帳戶的預設事件匯流排。
6. 針對 Rule type (規則類型)，選擇 Schedule (排程)。
7. 選擇 Next (下一步)。
8. 對於排程模式，請選擇以固定速率執行的排程，例如每 10 分鐘執行一次。 然後輸入 **5** 並從下拉式清單中選擇分鐘。
9. 選擇 Next (下一步)。
10. 在 Target types (目標類型) 欄位中，選擇 AWS service (服務)。
11. 對於選取目標，請從下拉式清單中選擇 Lambda 函數。
12. 在函數中，選取您在步驟 1：建立 Lambda 函數區段中建立的 Lambda 函數。在此範例中，選取 `LogScheduledEvent`。
13. 選擇 Next (下一步)。
14. 選擇 Next (下一步)。
15. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

建立規則 (AWS CLI)

1. 若要建立依排程執行的規則，請使用 `put-rule` 命令。

```
aws events put-rule \
--name my-scheduled-rule \
--schedule-expression 'rate(5 minutes)'
```

執行此規則時，會建立事件，然後將其傳送至目標。以下為範例 事件。

```
{
    "version": "0",
    "id": "53dc4d37-cffa-4f76-80c9-8b7d4a4d2eaa",
    "detail-type": "Scheduled Event",
    "source": "aws.events",
    "account": "123456789012",
    "time": "2015-10-08T16:53:06Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:events:us-east-1:123456789012:rule/my-scheduled-rule"
    ],
    "detail": {}
}
```

2. 若要授與 EventBridge 服務主體 (`events.amazonaws.com`) 權限來執行規則，請使用 `add-permission` 命令。

```
aws lambda add-permission \
--function-name LogScheduledEvent \
--statement-id my-scheduled-event \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/my-scheduled-rule
```

3. 使用下列內容建立檔案 `targets.json`。

```
[
  {
    "Id": "1",
    "Arn": "arn:aws:lambda:us-east-1:123456789012:function:LogScheduledEvent"
  }
```

```
]
```

4. 若要將您在步驟 1 中建立的 Lambda 函數新增至規則，請使用 `put-targets` 命令。

```
aws events put-targets --rule my-scheduled-rule --targets file://targets.json
```

步驟 3：驗證規則

步驟 2 完成後等待至少 5 分鐘，您可以驗證 Lambda 函數是否已被叫用。

檢視 Lambda 函數的輸出

1. 前往 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Logs (日誌)。
3. 為 Lambda 函數 (`/aws/lambda/function-name`) 選取日誌群組名稱。
4. 選取日誌串流的名稱以檢視函數為您啟動的執行個體所提供的資料。

步驟 4：確認成功

如果您在 CloudWatch logs 中看到 Lambda 事件，表示您已成功完成本教學課程。如果事件不在 CloudWatch logs 中，請驗證規則是否已成功建立，開始進行故障診斷，如果規則看起來正確，請驗證 Lambda 函數的程式碼是否正確。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

刪除 EventBridge 規則

1. 在 EventBridge 主控台中開啟規則頁面。
2. 選取您建立的規則。
3. 選擇刪除。
4. 選擇刪除。

若要刪除 Lambda 函數

1. 開啟 Lambda 主控台中的 函數頁面。
2. 選擇您建立的函數。
3. 選擇 動作、刪除。
4. 選擇 刪除。

與 SaaS 供應商整合的 Amazon EventBridge 教學課程

EventBridge 可以直接與 SaaS 合作夥伴應用程式和服務合作，以傳送和接收事件。以下教學課程說明如何將 EventBridge 整合到 SaaS 合作夥伴中。

教學：

- 教學課程：建立與 Datadog 的連線，作為 API 目的地
- 教學課程：建立與 Salesforce 的連線，作為 API 目的地
- 教學課程：建立 Zendesk 的連線，作為 API 目的地

教學課程：建立與 Datadog 的連線，作為 API 目的地

您可以使用 EventBridge 將事件路由傳送至第三方服務，例如：Datadog。

在本教學課程中，您將使用 EventBridge 主控台建立與 Datadog 的連線、指向 Datadog 的 API 目的地以及將事件路由傳送至 Datadog 的規則。

步驟：

- 先決條件
- 步驟 1：建立連線
- 步驟 2：建立 API 目的地
- 步驟 3：建立規則
- 步驟 4：測試規則
- 步驟 5：清理您的資源

先決條件

教學課程需要使用以下項目：

- 一個 Datadog 帳戶。
- 一個 Datadog API 金鑰。
- 一個已啟用 EventBridge 的 Amazon Simple Storage Service (Amazon S3) 儲存貯體

步驟 1：建立連線

若要將事件傳送至 Datadog，您必須先建立與 Datadog API 的連線。

建立連線

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 API 目的地。
3. 選擇連線標籤，然後選擇建立連線。
4. 輸入連線的名稱和描述。例如，輸入 **`Datadog`** 作為名稱並輸入 **`Datadog API Connection`** 作為描述。
5. 對於授權類型，請選擇 API 金鑰。

6. 對於 API 名稱，請輸入 **DD-API-KEY**。
7. 對於值，請貼上您的 Datadog 秘密 API 金鑰。
8. 選擇建立。

步驟 2：建立 API 目的地

由於您已建立連線，接下來您將建立要用作規則目標的 API 目的地。

建立 API 目的地

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 API 目的地。
3. 選擇建立 API 目的地。
4. 輸入 API 目的地的名稱與描述。例如，輸入 **DatadogAD** 作為名稱，**Datadog API Destination** 作為描述。
5. 對於 API 目標端點，請輸入 **https://http-intake.logs.datadoghq.com/api/v2/logs**。
6. 對於 HTTP method (HTTP 方法)，請選擇 POST。
7. 對於叫用率限制，請輸入 **300**。
8. 對於連線，請選擇使用現有連線，然後選擇您在步驟 1 中建立的 Datadog 連線。
9. 選擇建立。

步驟 3：建立規則

接下來，您將建立一個規則，在建立 Amazon S3 物件時，將事件傳送至 Datadog。

建立規則

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。例如，輸入 **DatadogRule** 作為名稱，**Rule to send events to Datadog for S3 object creation** 作為描述。
5. 針對 Event bus (事件匯流排) 選擇 default (預設值)。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。

7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 中，選擇 Other (其他)。
9. 針對事件模式，請執行下列動作：

```
{
  "source": ["aws.s3"]
}
```

10. 選擇 Next (下一步)。
11. 針對目標類型，請選擇 EventBridge API 目的地。
12. 針對 API 目的地，請選擇使用現有的 API 目的地，然後選擇您在步驟 2 中建立的 DatadogAD 目的地。
13. 針對執行角色，請選擇為此特定資源建立新角色。
14. 對於其他設定，請執行下列動作：
    a. 對於設定目標輸入，請從下拉式清單中選擇輸入轉換器。
    b. 選擇設定輸入轉換器。
    c. 針對範例事件，請輸入下列內容：

```
{
  "detail":[]
}
```

    d. 對於目標輸入轉換器，請執行下列動作：
        i. 對於輸入路徑，請輸入下列內容：

```
{"detail":"$.detail"}
```

        ii. 對於輸入範本，請輸入下列內容：

```
{"message": <detail>}
```

    e. 選擇確認。
15. 選擇 Next (下一步)。
16. 選擇 Next (下一步)。
17. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：測試規則

若要測試您的規則，請將檔案上傳到啟用 EventBridge 的儲存貯體，以建立 Amazon S3 物件。將建立的物件記錄在 Datadog 日誌主控台中。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

若要刪除 EventBridge 連線

1. 開啟 EventBridge 主控台的 API 目的地頁面。
2. 選擇 Connections (連線) 索引標籤。
3. 選取您建立的連線。
4. 選擇 Delete (刪除)。
5. 輸入連線名稱並選擇刪除。

若要刪除 EventBridge API 目的地

1. 開啟 EventBridge 主控台的 API 目的地頁面。
2. 選取您建立的 API 目的地。
3. 選擇 Delete (刪除)。
4. 輸入 API 目的地的名稱並選擇刪除。

刪除 EventBridge 規則

1. 開啟 EventBridge 主控台的 規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

教學課程：建立與 Salesforce 的連線，作為 API 目的地

您可以使用 EventBridge 將事件路由至第三方服務，例如Salesforce。

在本教學課程中，您將使用 EventBridge 主控台建立連線Salesforce、指向的 API 目的地Salesforce，以及將事件路由到的規則Salesforce。

步驟：

- 必要條件
- 步驟 1：建立連線
- 步驟 2：建立 API 目的地
- 步驟 3：建立規則
- 步驟 4：測試規則
- 步驟 5：清除您的資源

必要條件

教學課程需要使用以下項目：

- 一個 Salesforce 帳戶。
- Salesforce 已連線的應用程式。
- Salesforce 安全字符。
- Salesforce 自訂平台事件。
- 一個 EventBridge啟用的 Amazon Simple Storage Service (Amazon S3) 存儲桶。

步驟 1：建立連線

若要將事件傳送至 Salesforce，您必須先建立與 Salesforce API 的連線。

建立連線

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇 API 目的地。
3. 選擇連線標籤，然後選擇建立連線。

4. 輸入連線的名稱和描述。例如，輸入 **Salesforce** 作為名稱並輸入 **Salesforce API Connection** 作為描述。
5. 對于目的地類型，請選擇合作伙伴，針對合作伙伴目的地，從下拉式清單中選取 Salesforce。
6. 針對授權端點，輸入下列其中一個：
   - 如果您使用的是生產組織，請輸入 **https://*MyDomainName*.my.salesforce.com./services/oauth2/token**
   - 如果您使用的沙盒沒有增強網域，請輸入 **https://*MyDomainName*--*SandboxName*.my.salesforce.com/services /oauth2/token**
   - 如果您使用具有增強網域的沙盒，請輸入 **https://*MyDomainName*--*SandboxName*.sandbox.my.salesforce.com/services/oauth2/token**
7. 針對 HTTP 方法，請從下拉式清單中選擇 POST。
8. 針對用戶端 ID，請從 Salesforce 連線應用程式輸入用戶端 ID。
9. 針對用戶端密碼，請從 Salesforce 連線應用程式輸入用戶端密碼。
10. 針對 OAuth HTTP 參數，請輸入下列索引鍵/值組：

索引鍵	值
grant_type	client_credentials

11. 選擇建立。

步驟 2：建立 API 目的地

由於您已建立連線，接下來您將建立要用作規則目標的 API 目的地。

建立 API 目的地

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇 API 目的地。
3. 選擇建立 API 目的地。
4. 針對 API 目的地，輸入名稱和描述。例如，請輸入 **SalesforceAD** 作為名稱，**Salesforce API Destination** 作為描述。
5. 針對 API 目的地端點，請輸入 **https://*MyDomainName*.my.salesforce.com/services/data/v54.0/sobjects/*MyEvent__e***，其中 MyEvent__e 是您要在其中傳送資訊的平台事件。

6. 針對 HTTP 方法，請從下拉式清單中選擇 POST。
7. 針對調用率限制，请輸入 **300**。
8. 針對連線，選擇使用現有連線並選擇您在步驟 1 中建立的 Salesforce 連線。
9. 選擇建立。

步驟 3：建立規則

接下來，您將建立一個規則，在建立 Amazon S3 物件時，將事件傳送至 Salesforce。

建立規則

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在導覽窗格中，選擇規則。
3. 選擇建立規則。
4. 輸入規則的名稱和描述。例如，輸入 **SalesforceRule** 作為名稱，**Rule to send events to Salesforce for S3 object creation** 作為描述。
5. 針對事件匯流排選擇預設值。
6. 針對規則類型選擇具有事件模式的規則。
7. 選擇下一步。
8. 在事件來源中，選擇其他。
9. 針對事件模式，請輸入：

```
{
  "source": ["aws.s3"]
}
```

10. 選擇下一步。
11. 針對「目標類型」，選擇「EventBridge API 目標」。
12. 針對 API 目的地，請選擇使用現有的 API 目的地，然後選擇您在步驟 2 中建立的 SalesforceAD 目的地。
13. 針對執行角色，請選擇為此特定資源建立新角色。
14. 針對其他設定，請執行下列動作：

    a. 針對設定目標輸入，請從下拉式清單中選擇輸入轉換器。

    b. 選擇設定輸入轉換器。

c. 針對範例事件，請輸入以下內容：

```
{
  "detail":[]
}
```

d. 針對目標輸入轉換器，請執行下列動作：

i. 針對輸入路徑，請輸入以下內容：

```
{"detail":"$.detail"}
```

ii. 針對輸入範本，請輸入以下內容：

```
{"message": <detail>}
```

e. 選擇確認。

15. 選擇下一步。
16. 選擇下一步。
17. 檢閱規則的詳細資訊，然後選擇建立規則。

步驟 4：測試規則

若要測試您的規則，請將檔案上傳到 EventBridge已啟用的儲存貯體，以建立 Amazon S3 物件。有關建立物件的信息將被發送到 Salesforce 平台事件。

步驟 5：清除您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。刪除不再使用的 AWS 資源，即可避免 AWS 帳戶不必要的費用。

若要刪除 EventBridge 連線

1. 開啟主 EventBridge 控台的 API 目的地頁面。
2. 選擇 Connections (連線) 索引標籤。
3. 選取您建立的連線。
4. 選擇刪除。
5. 輸入連線名稱並選擇刪除。

若要刪除 EventBridge API 目的地

1. 開啟主 EventBridge 控台的 API 目的地頁面。
2. 選取您建立的 API 目的地。
3. 選擇刪除。
4. 輸入 API 目的地的名稱並選擇刪除。

若要刪除 EventBridge 規則

1. 開啟主 EventBridge 控台的 [規則] 頁面。
2. 選取您建立的規則。
3. 選擇刪除。
4. 選擇刪除。

教學課程：建立 Zendesk 的連線，作為 API 目的地

您可以使用 EventBridge 將 事件 路由傳送至第三方服務，例如 Zendesk。

在本教學課程中，您將使用 EventBridge 主控台建立 Zendesk 的連線、指向 Zendesk 的 API 目的地 以及將事件路由傳送至 Zendesk 的 規則。

步驟：

- 先決條件
- 步驟 1：建立連線
- 步驟 2：建立 API 目的地
- 步驟 3：建立規則
- 步驟 4：測試規則
- 步驟 5：清理您的資源

先決條件

教學課程需要使用以下項目：

- 一個 Zendesk 帳戶。
- 一個啟用 EventBridge 的 Amazon Simple Storage Service (Amazon S3) 儲存貯體

步驟 1：建立連線

若要將事件傳送至 Zendesk，您必須先建立與 Zendesk API 的連線。

建立連線

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 API 目的地。
3. 選擇 連線 標籤，然後選擇 建立連線。
4. 輸入連線的名稱和描述。在此範例中，請輸入 **Zendesk** 作為名稱，**Connection to Zendesk API** 作為描述。
5. 對於 授權類型，選擇 基本 (使用者名稱/密碼)。
6. 對於 使用者名稱，輸入您的 Zendesk 使用者名稱。

7. 對於 密碼，輸入您的 Zendesk 密碼。
8. 選擇 Create (建立)。

步驟 2：建立 API 目的地

由於您已建立連線，接下來您將建立 API 目的地，以用作規則的 目標。

建立 API 目的地

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 API 目的地。
3. 選擇 建立 API 目的地。
4. 針對 API 目的地，輸入名稱和描述。在此範例中，請輸入 **ZendeskAD** 作為名稱，**Zendesk API destination** 作為描述。
5. 對於 API 目標端點，請輸入 **https://*your-subdomain*.zendesk.com/api/v2/tickets.json**，其中 ***#####*** 是與您 Zendesk 帳戶關聯的子網域。
6. 針對 HTTP method (HTTP 方法)，選擇 POST。
7. 針對 調用限制，輸入 **10**。
8. 針對 連線，選擇 使用現有連線 並選擇您在步驟 1 中建立的 Zendesk 連線。
9. 選擇 Create (建立)。

步驟 3：建立規則

接下來，建立一個規則，在建立 Amazon S3 物件時，將事件傳送至 Zendesk。

建立規則

1. 前往 https://console.aws.amazon.com/events/ 開啟 Amazon EventBridge 主控台。
2. 在導覽窗格中，選擇 Rules (規則)。
3. 選擇 Create rule (建立規則)。
4. 輸入規則的名稱和描述。在此範例中，請輸入 **ZendeskRule** 作為名稱，**Rule to send events to Zendesk when S3 objects are created** 作為描述。
5. 針對 Event bus (事件匯流排) 選擇 default (預設值)。
6. 針對 Rule type (規則類型) 選擇 Rule with an event pattern (具有事件模式的規則)。

7. 選擇 Next (下一步)。
8. 在 Event source (事件來源) 中，選擇 Other (其他)。
9. 針對 事件模式，請執行下列動作：

```
{
  "source": ["aws.s3"]
}
```

10. 選擇 Next (下一步)。
11. 針對 目標類型，請選擇 EventBridge API 目的地。
12. 針對 API 目的地，請選擇 使用現有的 API 目的地，然後選擇您在步驟 2 中建立的 ZendeskAD 目的地。
13. 針對 執行角色，請選擇 為此特定資源建立新角色。
14. 對於 其他設定，執行下列動作：
    a. 對於 設定目標輸入，請從下拉式清單中選擇 輸入轉換器。
    b. 選擇 設定輸入轉換器。
    c. 針對 範例事件，輸入以下內容：

```
{
  "detail":[]
}
```

    d. 對於 目標輸入轉換器，請執行下列動作：
       i. 對於 輸入路徑，輸入以下內容：

```
{"detail":"$.detail"}
```

       ii. 對於 輸入範本，輸入以下內容：

```
{"message": <detail>}
```

    e. 選擇 確認。
15. 選擇 Next (下一步)。
16. 選擇 Next (下一步)。
17. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

步驟 4：測試規則

若要測試您的規則，請將檔案上傳到啟用 EventBridge 的儲存貯體，以建立 Amazon S3 物件。當事件符合規則時，EventBridge 會呼叫 Zendesk 建立票證 API。新票證將出現在 Zendesk 儀表板中。

步驟 5：清理您的資源

除非您想要保留為此教學課程建立的資源，否則您現在便可刪除。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶避免不必要的費用。

若要刪除 EventBridge 連線

1. 開啟 EventBridge 主控台的 API 目的地頁面。
2. 選擇 Connections (連線) 索引標籤。
3. 選取您建立的連接。
4. 選擇 Delete (刪除)。
5. 輸入連線名稱並選擇 刪除。

若要刪除 EventBridge API 目的地

1. 開啟 EventBridge 主控台的 API 目的地頁面。
2. 選取您建立的 API 目的地。
3. 選擇 Delete (刪除)。
4. 輸入 API 目的地的名稱並選擇 刪除。

刪除 EventBridge 規則

1. 在 Amazon EventBridge 主控台中開啟 規則頁面。
2. 選取您建立的規則。
3. 選擇 Delete (刪除)。
4. 選擇 Delete (刪除)。

搭 EventBridge 配 AWS SDK 使用

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

如需特定的範例 EventBridge，請參閱 EventBridge 使用 AWS SDK 的程式碼範例。

> **可用性範例**
>
> 找不到所需的內容嗎？ 請使用本頁面底部的提供意見回饋連結申請程式碼範例。

EventBridge 使用 AWS SDK 的程式碼範例

下列程式碼範例顯示如何搭 EventBridge 配 AWS 軟體開發套件 (SDK) 使用。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

Cross-service examples (跨服務範例) 是跨多個 AWS 服務執行的應用程式範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

開始使用

你好 EventBridge

下列程式碼範例會示範如何開始使用 EventBridge。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using Amazon.EventBridge;
using Amazon.EventBridge.Model;

namespace EventBridgeActions;

public static class HelloEventBridge
{
    static async Task Main(string[] args)
    {
        var eventBridgeClient = new AmazonEventBridgeClient();
```

```
        Console.WriteLine($"Hello Amazon EventBridge! Following are some of your
 EventBuses:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five event buses.
        var response = await eventBridgeClient.ListEventBusesAsync(
            new ListEventBusesRequest()
            {
                Limit = 5
            });

        foreach (var eventBus in response.EventBuses)
        {
            Console.WriteLine($"\tEventBus: {eventBus.Name}");
            Console.WriteLine($"\tArn: {eventBus.Arn}");
            Console.WriteLine($"\tPolicy: {eventBus.Policy}");
            Console.WriteLine();
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListEventBuses中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " +
 bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListEventBuses中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request = ListEventBusesRequest {
        limit = 10
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response: ListEventBusesResponse =
 eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListEventBuses中的 Kotlin API 參考。

程式碼範例

- EventBridge 使用 AWS SDK 的動作
  - 搭DeleteRule配 AWS SDK 或命令列工具使用
  - 搭DescribeRule配 AWS SDK 或命令列工具使用

- 搭DisableRule配 AWS SDK 或命令列工具使用
- 搭EnableRule配 AWS SDK 或命令列工具使用
- 搭ListRuleNamesByTarget配 AWS SDK 或命令列工具使用
- 搭ListRules配 AWS SDK 或命令列工具使用
- 搭ListTargetsByRule配 AWS SDK 或命令列工具使用
- 搭PutEvents配 AWS SDK 或命令列工具使用
- 搭PutRule配 AWS SDK 或命令列工具使用
- 搭PutTargets配 AWS SDK 或命令列工具使用
- 搭RemoveTargets配 AWS SDK 或命令列工具使用

- EventBridge 使用 AWS SDK 的案例
  - EventBridge 使用開發 AWS 套件在 Amazon 中建立和觸發規則
  - 使用 AWS SDK 開始使用 EventBridge 規則和目標
- 使用 SDK 的跨服務 EventBridge 範 AWS 例
  - 使用排程事件來調用 Lambda 函數

EventBridge 使用 AWS SDK 的動作

下列程式碼範例示範如何使用 AWS SDK 執 EventBridge行個別動作。這些摘錄會呼叫 EventBridge API，是來自必須在內容中執行的大型程式碼摘錄。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執行程式碼的指示。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 Amazon EventBridge API 參考資料。

範例

- 搭DeleteRule配 AWS SDK 或命令列工具使用
- 搭DescribeRule配 AWS SDK 或命令列工具使用
- 搭DisableRule配 AWS SDK 或命令列工具使用
- 搭EnableRule配 AWS SDK 或命令列工具使用
- 搭ListRuleNamesByTarget配 AWS SDK 或命令列工具使用
- 搭ListRules配 AWS SDK 或命令列工具使用
- 搭ListTargetsByRule配 AWS SDK 或命令列工具使用
- 搭PutEvents配 AWS SDK 或命令列工具使用

- 搭PutRule配 AWS SDK 或命令列工具使用
- 搭PutTargets配 AWS SDK 或命令列工具使用
- 搭RemoveTargets配 AWS SDK 或命令列工具使用

搭**DeleteRule**配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用DeleteRule。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

根據其名稱刪除規則。

```
/// <summary>
/// Delete an event rule by name.
/// </summary>
/// <param name="ruleName">The name of the event rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteRuleByName(string ruleName)
{
    var response = await _amazonEventBridge.DeleteRuleAsync(
        new DeleteRuleRequest()
        {
            Name = ruleName
        });

    return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteRule中的。

CLI

AWS CLI

刪除 CloudWatch 事件規則

此範例會刪除名為 EC2 的規則InstanceStateChanges：

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteRule中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteRule中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest = DeleteRuleRequest {
        name = ruleName
    }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteRule中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DescribeRule`配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用`DescribeRule`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用規則描述取得規則的狀態。

```
    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DescribeRule中的。

CLI

AWS CLI

顯示 CloudWatch 事件規則的相關資訊

此範例顯示名為下列規則的相關資訊 DailyLambdaFunction：

```
aws events describe-rule --name "DailyLambdaFunction"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeRule中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DescribeRule中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest = DescribeRuleRequest {
        name = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DescribeRule中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DisableRule**配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用`DisableRule`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

根據其規則名稱停用規則。

```
/// <summary>
/// Disable a particular rule on an event bus.
/// </summary
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DisableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
        new DisableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DisableRule中的。

CLI

AWS CLI

停用 CloudWatch 事件規則

此範例會停用名為的規則 DailyLambdaFunction。此規則不會遭到刪除：

```
aws events disable-rule --name "DailyLambdaFunction"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DisableRule中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用其規則名稱停用規則。

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DisableRule中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun changeRuleState(eventRuleName: String, isEnabled: Boolean?) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest = DisableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest = EnableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DisableRule中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`EnableRule`配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用`EnableRule`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

根據其規則名稱啟用規則。

```
/// <summary>
/// Enable a particular rule on an event bus.
/// </summary>
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
        new EnableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考EnableRule中的。

CLI

AWS CLI

啟用 CloudWatch 事件規則

此範例會啟用先前已停用名為 DailyLambdaFunction的規則：

```
aws events enable-rule --name "DailyLambdaFunction"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考EnableRule中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用其規則名稱啟用規則。

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考EnableRule中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun changeRuleState(eventRuleName: String, isEnabled: Boolean?) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest = DisableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest = EnableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK EnableRule中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListRuleNamesByTarget配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用ListRuleNamesByTarget。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

列出使用目標的所有規則名稱。

```
    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
_amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;
```

```
        } while (response.NextToken is not null);

        return results;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListRuleNamesByTarget中的。

CLI

AWS CLI

顯示具有指定目標的所有規則

此範例顯示以名為 "MyFunctionName" 作為目標的 Lambda 函數的所有規則：

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-
east-1:123456789012:function:MyFunctionName"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListRuleNamesByTarget中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用目標列出所有規則名稱。

```
   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();
```

```
        ListRuleNamesByTargetResponse response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListRuleNamesByTarget中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest = ListRuleNamesByTargetRequest {
        targetArn = topicArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListRuleNamesByTarget中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListRules配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用ListRules。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

列出事件匯流排的所有規則。

```
    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;
```

```
        } while (response.NextToken is not null);

        return results;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListRules中的。

CLI

AWS CLI

顯示所有 CloudWatch 事件規則的清單

此範例顯示區域中的所有 CloudWatch 事件規則：

```
aws events list-rules
```

顯示以特定字串開頭的 CloudWatch 事件規則清單。

此範例顯示區域中名稱以「Daily」開頭的所有 CloudWatch 事件規則：

```
aws events list-rules --name-prefix "Daily"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListRules中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用其規則名稱啟用規則。

```
    public static void listRules(EventBridgeClient eventBrClient) {
```

```
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " +
 rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[ListRules](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun listRules() {
    val rulesRequest = ListRulesRequest {
        eventBusName = "default"
        limit = 10
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
```

```
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListRules中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListTargetsByRule`配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用`ListTargetsByRule`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用規則名稱列出規則的所有目標。

```
    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
```

```
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListTargetsByRule中的。

CLI

AWS CLI

顯示 CloudWatch 事件規則的所有目標

此範例顯示名為下列規則的所有目標 DailyLambdaFunction：

```
aws events list-targets-by-rule --rule  "DailyLambdaFunction"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListTargetsByRule中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用規則名稱列出規則的所有目標。

```
    public static void listTargets(EventBridgeClient eventBrClient, String
 ruleName) {
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
 eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListTargetsByRule中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest = ListTargetsByRuleRequest {
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListTargetsByRule中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutEvents配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用PutEvents。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 建立和觸發規則
- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

傳送符合規則之自訂模式的事件。

```
    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
```

```
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutEvents中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

傳送事件。

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
        event_entry.SetDetail(MakeDetails(event_key, event_value));
        event_entry.SetDetailType("sampleSubmitted");
        event_entry.AddResources(resource_arn);
        event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

        Aws::CloudWatchEvents::Model::PutEventsRequest request;
        request.AddEntries(event_entry);

        auto outcome = cwe.PutEvents(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to post CloudWatch event: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully posted CloudWatch event" << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutEvents中的。

CLI

AWS CLI

將自訂事件傳送至 CloudWatch 事件

此範例會將自訂事件傳送至 CloudWatch 事件。putevents.json 檔案中包含該事件：

```
aws events put-events --entries file://putevents.json
```

以下為 putevents.json 檔案的內容：

```
[
```

```
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value1\", \"key2\": \"value2\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
  },
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value3\", \"key2\": \"value4\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
   }
]
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutEvents](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
```

```
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutEvents中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import {
  EventBridgeClient,
  PutEventsCommand,
} from "@aws-sdk/client-eventbridge";

export const putEvents = async (
  source = "eventbridge.integration.test",
  detailType = "greeting",
  resources = [],
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutEventsCommand({
      Entries: [
        {
```

```
          Detail: JSON.stringify({ greeting: "Hello there." }),
          DetailType: detailType,
          Resources: resources,
          Source: source,
        },
      ],
    }),
  );

  console.log("PutEvents response:");
  console.log(response);
  // PutEvents response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3d0df73d-dcea-4a23-ae0d-f5556a3ac109',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Entries: [ { EventId: '51620841-5af4-6402-d9bc-b77734991eb5' } ],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutEvents中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

ebevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutEvents中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun triggerCustomRule(email: String) {
    val json = "{" +
        "\"UserEmail\": \"" + email + "\"," +
        "\"Message\": \"This event was generated by example code.\"" +
        "\"UtcTime\": \"Now.\"" +
        "}"
```

```
    val entry = PutEventsRequestEntry {
        source = "ExampleSource"
        detail = json
        detailType = "ExampleType"
    }

    val eventsRequest = PutEventsRequest {
        this.entries = listOf(entry)
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutEvents中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutRule配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用PutRule。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 建立和觸發規則
- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立在物件新增至 Amazon Simple Storage Service 儲存貯體時觸發的規則。

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                               "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }
```

建立使用自訂模式的規則。

```
    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
```

```
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutRule中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

建立 規則。

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;
        Aws::CloudWatchEvents::Model::PutRuleRequest request;
        request.SetName(rule_name);
        request.SetRoleArn(role_arn);
        request.SetScheduleExpression("rate(5 minutes)");
        request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

        auto outcome = cwe.PutRule(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events rule " <<
                rule_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch events rule " <<
                rule_name << " with resulting Arn " <<
                outcome.GetResult().GetRuleArn() << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutRule中的。

CLI

AWS CLI

建立 CloudWatch 事件規則

此範例會建立規則，該規則會在每天 UTC 時間上午 9:00 時觸發。如果您使用 put-targets，新增 Lambda 函數作為此規則的目標，則可以在指定的時間每天執行 Lambda 函數：

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9
 * * ? *)"
```

此範例會建立規則，當區域中的任何 EC2 執行個體變更狀態時便會觸發此規則：

```
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{\"source
\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
 --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

此範例會建立規則，當區域中的任何 EC2 執行個體停止或終止時便會觸發此規則：

```
aws events put-rule --name "EC2InstanceStateChangeStopOrTerminate" --event-
pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change
 Notification\"],\"detail\":{\"state\":[\"stopped\",\"terminated\"]}}" --role-arn
 "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutRule中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立排程規則。

```
   public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
       try {
           PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by
the Java API")
                   .build();

           PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
           System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());
```

```
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

建立在物件新增至 Amazon Simple Storage Service 儲存貯體時觸發的規則。

```
    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutRule中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { EventBridgeClient, PutRuleCommand } from "@aws-sdk/client-eventbridge";

export const putRule = async (
  ruleName = "some-rule",
  source = "some-source",
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutRuleCommand({
      Name: ruleName,
      EventPattern: JSON.stringify({ source: [source] }),
      State: "ENABLED",
      EventBusName: "default",
    }),
  );

  console.log("PutRule response:");
  console.log(response);
  // PutRule response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd7292ced-1544-421b-842f-596326bc7072',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
```

```
  //   },
  //   RuleArn: 'arn:aws:events:us-east-1:xxxxxxxxxxxx:rule/
EventBridgeTestRule-1696280037720'
  // }
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutRule中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

ebevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutRule中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立排程規則。

```
suspend fun createScRule(ruleName: String?, cronExpression: String?) {
    val ruleRequest = PutRuleRequest {
        name = ruleName
        eventBusName = "default"
        scheduleExpression = cronExpression
        state = RuleState.Enabled
        description = "A test rule that runs on a schedule created by the Kotlin
 API"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

建立在物件新增至 Amazon Simple Storage Service 儲存貯體時觸發的規則。

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(roleArnVal: String?, bucketName: String, eventRuleName:
 String?) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
```

```
        }
    }"""

    val ruleRequest = PutRuleRequest {
        description = "Created by using the AWS SDK for Kotlin"
        name = eventRuleName
        eventPattern = pattern
        roleArn = roleArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutRule中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutTargets配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用PutTargets。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用規則和目標

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

新增作爲某個規則目標的 Amazon SNS 主題。

```
   /// <summary>
   /// Add an Amazon SNS target topic to a rule.
   /// </summary>
   /// <param name="ruleName">The name of the rule to update.</param>
   /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
   /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
   /// <returns>The ID of the target.</returns>
   public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
   {
       var targetID = Guid.NewGuid().ToString();

       // Create the list of targets and add a new target.
       var targets = new List<Target>
       {
           new Target()
           {
               Arn = targetArn,
               Id = targetID
           }
       };

       // Add the targets to the rule.
       var response = await _amazonEventBridge.PutTargetsAsync(
           new PutTargetsRequest()
           {
               EventBusName = eventBusArn,
               Rule = ruleName,
               Targets = targets,
           });

       if (response.FailedEntryCount > 0)
       {
           response.FailedEntries.ForEach(e =>
           {
               _logger.LogError(
                   $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
           });
       }
```

```
        return targetID;
    }
```

將輸入轉換器新增至某個規則的目標。

```
    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
```

```
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutTargets中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

新增目標。

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::Target target;
```

```
        target.SetArn(lambda_arn);
        target.SetId(target_id);

        Aws::CloudWatchEvents::Model::PutTargetsRequest request;
        request.SetRule(rule_name);
        request.AddTargets(target);

        auto putTargetsOutcome = cwe.PutTargets(request);
        if (!putTargetsOutcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout <<
                "Successfully created CloudWatch events target for rule "
                << rule_name << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutTargets中的。

CLI

AWS CLI

若要新增 CloudWatch 事件規則的目標

此範例會新增 Lambda 函數作為規則的目標：

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

此範例會將 Amazon Kinesis 串流設定為目標，以便將此規則捕捉到的事件轉送至串流：

```
aws events put-targets --rule EC2InstanceStateChanges --targets
 "Id"="1","Arn"="arn:aws:kinesis:us-east-1:123456789012:stream/
MyStream","RoleArn"="arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

此範例會將兩個 Amazon Kinesis 串流設定為單一規則的目標：

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="Target1","Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream1","RoleArn"="arn:aws:iam::379642911888:role/ MyRoleToAccessLambda"
  "Id"="Target2"," Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream2","RoleArn"="arn:aws:iam::379642911888:role/MyRoleToAccessLambda"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutTargets中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

新增作爲某個規則目標的 Amazon SNS 主題。

```
    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
```

```
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }
```

將輸入轉換器新增至某個規則的目標。

```
    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutTargets中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import {
  EventBridgeClient,
  PutTargetsCommand,
} from "@aws-sdk/client-eventbridge";

export const putTarget = async (
  existingRuleName = "some-rule",
  targetArn = "arn:aws:lambda:us-east-1:000000000000:function:test-func",
  uniqueId = Date.now().toString(),
) => {
  const client = new EventBridgeClient({});
  const response = await client.send(
    new PutTargetsCommand({
      Rule: existingRuleName,
      Targets: [
        {
          Arn: targetArn,
          Id: uniqueId,
        },
      ],
    }),
  );

  console.log("PutTargets response:");
  console.log(response);
  // PutTargets response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'f5b23b9a-2c17-45c1-ad5c-f926c3692e3d',
  //     extendedRequestId: undefined,
```

```
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   FailedEntries: [],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutTargets中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myEventBridgeTarget",
    },
  ],
};

ebevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Success", data);
  }
});
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutTargets中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(ruleName: String?, topicArn: String?, topicName:
 String, eventRuleName: String, bucketName: String) {
    val targetID = UUID.randomUUID().toString()
    val myTarget = Target {
        id = targetID
        arn = topicArn
    }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request = PutTargetsRequest {
        eventBusName = null
        targets = targetsOb
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
```

```
}
```

將輸入轉換器新增至某個規則的目標。

```
suspend fun updateCustomRuleTargetWithTransform(topicArn: String?, ruleName:
 String?) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb = InputTransformer {
        inputTemplate = "\"Notification: sample event was received.\""
    }

    val target = Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransformerOb
    }

    val targetsRequest = PutTargetsRequest {
        rule = ruleName
        targets = listOf(target)
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutTargets中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**RemoveTargets**配 AWS SDK 或命令列工具使用

下列程式碼範例會示範如何使用RemoveTargets。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用規則名稱移除規則的所有目標。

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
```

```
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考RemoveTargets中的。

CLI

AWS CLI

移除事件的目標

此範例會將名為 MyStream 1 的 Amazon Kinesis 串流移除成為規則 DailyLambdaFunction的目標。建立 DailyLambdaFunction 時，此串流會設定為識別碼為 Target1 的目標：

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考RemoveTargets中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用規則名稱移除規則的所有目標。

```
    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
```

```
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[RemoveTargets](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request = ListTargetsByRuleRequest {
        rule = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
```

```
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest = RemoveTargetsRequest {
                    rule = eventRuleName
                    ids = listOf(myTarget.id.toString())
                }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK RemoveTargets中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

EventBridge 使用 AWS SDK 的案例

下列程式碼範例說明如何在 AWS SDK 中 EventBridge實作常見案例。這些案例會示範如何透過在其中呼叫多個函式來完成特定工作 EventBridge。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

範例

- EventBridge 使用開發 AWS 套件在 Amazon 中建立和觸發規則
- 使用 AWS SDK 開始使用 EventBridge 規則和目標

EventBridge 使用開發 AWS 套件在 Amazon 中建立和觸發規則

下列程式碼範例顯示如何在 Amazon 中建立和觸發規則 EventBridge。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

以正確的順序呼叫函數。

```
require "aws-sdk-sns"
require "aws-sdk-iam"
require "aws-sdk-cloudwatchevents"
require "aws-sdk-ec2"
require "aws-sdk-cloudwatch"
require "aws-sdk-cloudwatchlogs"
require "securerandom"
```

檢查提供給此函數的主題中是否存在指定的 Amazon Simple Notification Service (Amazon SNS) 主題。

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end

def topic_found?(topics, topic_arn)
```

```
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end
```

檢查 Amazon SNS 中呼叫者可用的主題中是否存在指定的主題。

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts "Topic found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
```

```
end
```

在 Amazon SNS 中建立主題，然後訂閱電子郵件地址以接收該主題的通知。

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

檢查提供給此函數的角色中是否存在指定的 AWS Identity and Access Management （IAM）角色。

```
# Checks whether the specified AWS Identity and Access Management (IAM)
```

```
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

檢查 IAM 中呼叫者可用的角色中是否存在指定的角色。

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
```

```
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts "Role found."
          return true
        end
      end
    end
  end
  puts "Role not found."
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end
```

在 IAM 中建立角色。

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "",
          'Effect': "Allow",
```

```
          'Principal': {
            'Service': "events.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }.to_json,
    path: "/",
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts "Adding access policy to role..."
  iam_client.put_role_policy(
    policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "CloudWatchEventsFullAccess",
          'Effect': "Allow",
          'Resource': "*",
          'Action': "events:*"
        },
        {
          'Sid': "IAMPassRoleForCloudWatchEvents",
          'Effect': "Allow",
          'Resource': "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
          'Action': "iam:PassRole"
        }
      ]
    }.to_json,
    policy_name: "CloudWatchEventsPolicy",
    role_name: role_name
  )
  puts "Access policy added to role."
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts "If the role was created, you must add the access policy " \
    "to the role yourself, or delete the role yourself and try again."
  return "Error"
end
```

檢查指定的 EventBridge 規則是否存在於那些提供給此函數之間。

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  return false
end
```

檢查指定的規則是否存在於那些可供呼叫者 EventBridge。

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
```

```
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts "Rule found."
          return true
        end
      end
    end
  end
  puts "Rule not found."
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end
```

在中建立規則 EventBridge。

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
```

```
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        "aws.ec2"
      ],
      'detail-type': [
        "EC2 Instance State-change Notification"
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: "ENABLED",
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
```

```
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts "If the rule was created, you must add the target " \
    "to the rule yourself, or delete the rule yourself and try again."
  return false
end
```

檢查 Amazon CloudWatch Logs 中呼叫者可用的記錄群組中是否存在指定的日誌群組。

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
```

```
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts "Log group found."
        return true
      end
    end
  end
  puts "Log group not found."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end
```

在記錄檔中建立 CloudWatch 記錄群組。

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts "Log group created."
  return true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  return false
end
```

在記錄檔中將事件寫入 CloudWatch 記錄資料流。

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
```

```
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

  response = cloudwatchlogs_client.put_log_events(event)
  puts "Message logged."
  return response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

重新啟動 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，並將相關活動的相關資訊新增至 CloudWatch 日誌中的日誌串流。

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
```

```
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ""

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    "This might take a few minutes..."
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts "Instance stopped."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts "Attempting to restart the instance. This might take a few minutes..."
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts "Instance restarted."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  return true
rescue StandardError => e
  puts "Error creating log stream or stopping or restarting the instance: " \
    "#{e.message}"
  log_event(
```

```
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  return false
end
```

顯示中規則活動的相關資訊 EventBridge。

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
```

```
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      "specified time period."
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

顯示記錄檔群組中所有記錄資料流的 CloudWatch 記錄資訊。

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
```

```
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts "Attempting to display log stream data for the log group " \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: "LastEventTime",
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
      describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts "-" * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts "No log messages for this log stream."
      end
    end
  end
rescue StandardError => e
  puts "Error getting information about the log streams or their messages: " \
    "#{e.message}"
end
```

向來電者顯示提醒，以手動清除他們不再需要的任何相關 AWS 資源。

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
```

```
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log
 group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts "-" * 10
  puts "Some of the following AWS resources might still exist in your account."
  puts "If you no longer want to use this code example, then to clean up"
  puts "your AWS account and avoid unexpected costs, you might want to"
  puts "manually delete any of the following resources if they exist:"
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

# Example usage:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = "aws-doc-sdk-examples-topic"
  email_address = "mary@example.com"
  # Properties for the IAM role.
  role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"
  # Properties for the Amazon EventBridge rule.
  rule_name = "aws-doc-sdk-examples-ec2-state-change"
  rule_description = "Triggers when any available EC2 instance starts."
  instance_state = "running"
  target_id = "sns-topic"
  # Properties for the Amazon EC2 instance.
  instance_id = "i-033c48ef067af3dEX"
  # Properties for displaying the event rule's activity.
  start_time = Time.now - 600 # Go back over the past 10 minutes
```

```
                                # (10 minutes * 60 seconds = 600 seconds).
end_time = Time.now
period = 60 # Look back every 60 seconds over the past 10 minutes.
# Properties for the Amazon CloudWatch Logs log group.
log_group_name = "aws-doc-sdk-examples-cloudwatch-log"
# AWS service clients for this code example.
region = "us-east-1"
sts_client = Aws::STS::Client.new(region: region)
sns_client = Aws::SNS::Client.new(region: region)
iam_client = Aws::IAM::Client.new(region: region)
cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
ec2_client = Aws::EC2::Client.new(region: region)
cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

# Get the caller's account ID for use in forming
# Amazon Resource Names (ARNs) that this code relies on later.
account_id = sts_client.get_caller_identity.account

# If the Amazon SNS topic doesn't exist, create it.
topic_arn = "arn:aws:sns:#{region}:#{account_id}:#{topic_name}"
unless topic_exists?(sns_client, topic_arn)
  topic_arn = create_topic(sns_client, topic_name, email_address)
  if topic_arn == "Error"
    puts "Could not create the Amazon SNS topic correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
    exit 1
  end
end

# If the IAM role doesn't exist, create it.
role_arn = "arn:aws:iam::#{account_id}:role/#{role_name}"
unless role_exists?(iam_client, role_arn)
  role_arn = create_role(iam_client, role_name)
  if role_arn == "Error"
    puts "Could not create the IAM role correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end
```

```
# If the Amazon EventBridge rule doesn't exist, create it.
unless rule_exists?(cloudwatchevents_client, rule_name)
  unless rule_created?(
    cloudwatchevents_client,
    rule_name,
    rule_description,
    instance_state,
    role_arn,
    target_id,
    topic_arn
  )
    puts "Could not create the Amazon EventBridge rule correctly. " \
      "Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts "Could not create the Amazon CloudWatch Logs log group " \
    "correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# Restart the Amazon EC2 instance, which triggers the rule.
unless instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  puts "Could not restart the instance to trigger the rule. " \
    "Continuing anyway to show information about the rule and logs..."
end

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity(
  cloudwatch_client,
```

```
    rule_name,
    start_time,
    end_time,
    period
  )

  # Display related log data in Amazon CloudWatch Logs.
  display_log_data(cloudwatchlogs_client, log_group_name)

  # Reminder the caller to clean up any AWS resources that are used
  # by this code example and are no longer needed.
  manual_cleanup_notice(
    topic_name, role_name, rule_name, log_group_name, instance_id
  )
end

run_me if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Ruby API 參考》中的下列主題。
  - PutEvents
  - PutRule

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 開始使用 EventBridge 規則和目標

下列程式碼範例示範如何：

- 建立規則並在其中新增目標。
- 啟用和停用規則。
- 列出並更新規則和目標。
- 發送事件，然後清理資源。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

在命令提示中執行互動式案例。

```
public class EventBridgeScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks with Amazon EventBridge:
    - Create a rule.
    - Add a target to a rule.
    - Enable and disable rules.
    - List rules and targets.
    - Update rules and targets.
    - Send events.
    - Delete the rule.
    */

    private static ILogger logger = null!;
    private static EventBridgeWrapper _eventBridgeWrapper = null!;
    private static IConfiguration _configuration = null!;

    private static IAmazonIdentityManagementService? _iamClient = null!;
    private static IAmazonSimpleNotificationService? _snsClient = null!;
    private static IAmazonS3 _s3Client = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonEventBridge>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonS3>()
            .AddAWSService<IAmazonSimpleNotificationService>()
            .AddTransient<EventBridgeWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<EventBridgeScenario>();

        ServicesSetup(host);

        string topicArn = "";
        string roleArn = "";

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            roleArn = await CreateRole();

            await CreateBucketWithEventBridgeEvents();

            await AddEventRule(roleArn);

            await ListEventRules();

            topicArn = await CreateSnsTopic();
```

```
            var email = await SubscribeToSnsTopic(topicArn);

            await AddSnsTarget(topicArn);

            await ListTargets();

            await ListRulesForTarget(topicArn);

            await UploadS3File(_s3Client);

            await ChangeRuleState(false);

            await GetRuleState();

            await UpdateSnsEventRule(topicArn);

            await ChangeRuleState(true);

            await UploadS3File(_s3Client);

            await UpdateToCustomRule(topicArn);

            await TriggerCustomRule(email);

            await CleanupResources(topicArn);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources(topicArn);
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon EventBridge example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
```

```
        _eventBridgeWrapper =
host.Services.GetRequiredService<EventBridgeWrapper>();
        _snsClient =
host.Services.GetRequiredService<IAmazonSimpleNotificationService>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    }

    /// <summary>
    /// Create a role to be used by EventBridge.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateRole()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating a role to use with EventBridge and attaching
managed policy AmazonEventBridgeFullAccess.");
        Console.WriteLine(new string('-', 80));

        var roleName = _configuration["roleName"];

        var assumeRolePolicy = "{" +
                               "\"Version\": \"2012-10-17\"," +
                               "\"Statement\": [{" +
                               "\"Effect\": \"Allow\"," +
                               "\"Principal\": {" +
                               $"\"Service\": \"events.amazonaws.com\"" +
                               "}," +
                               "\"Action\": \"sts:AssumeRole\"" +
                               "}]" +
                               "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = roleName
            });

        await _iamClient.AttachRolePolicyAsync(
            new AttachRolePolicyRequest()
            {
```

```
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
                RoleName = roleName
            });
        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Create an Amazon Simple Storage Service (Amazon S3) bucket with
 EventBridge events enabled.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateBucketWithEventBridgeEvents()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an S3 bucket with EventBridge events
 enabled.");

        var testBucketName = _configuration["testBucketName"];

        var bucketExists = await
 Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            testBucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = testBucketName,
                UseClientRegion = true
            });
        }

        await _s3Client.PutBucketNotificationAsync(new
 PutBucketNotificationRequest()
        {
            BucketName = testBucketName,
            EventBridgeConfiguration = new EventBridgeConfiguration()
        });

        Console.WriteLine($"\tAdded bucket {testBucketName} with EventBridge
 events enabled.");
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and upload a file to an S3 bucket to trigger an event.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UploadS3File(IAmazonS3 s3Client)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Uploading a file to the test bucket. This will trigger
a subscription email.");

        var testBucketName = _configuration["testBucketName"];

        var fileName = $"example_upload_{DateTime.UtcNow.Ticks}.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for testing uploads.");
        }

        await s3Client.PutObjectAsync(new PutObjectRequest()
        {
            FilePath = fileName,
            BucketName = testBucketName
        });

        Console.WriteLine($"\tPress Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an Amazon Simple Notification Service (Amazon SNS) topic to use as
an EventBridge target.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> CreateSnsTopic()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "Creating an Amazon Simple Notification Service (Amazon SNS) topic
for email subscriptions.");

        var topicName = _configuration["topicName"];

        string topicPolicy = "{" +
                             "\"Version\": \"2012-10-17\"," +
                             "\"Statement\": [{" +
                             "\"Sid\": \"EventBridgePublishTopic\"," +
                             "\"Effect\": \"Allow\"," +
                             "\"Principal\": {" +
                             $"\"Service\": \"events.amazonaws.com\"" +
                             "}," +
                             "\"Resource\": \"*\"," +
                             "\"Action\": \"sns:Publish\"" +
                             "}]" +
                             "}";

        var topicAttributes = new Dictionary<string, string>()
        {
            { "Policy", topicPolicy }
        };

        var topicResponse = await _snsClient!.CreateTopicAsync(new
CreateTopicRequest()
        {
            Name = topicName,
            Attributes = topicAttributes

        });

        Console.WriteLine($"\tAdded topic {topicName} for email subscriptions.");

        Console.WriteLine(new string('-', 80));

        return topicResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a user email to an SNS topic.
    /// </summary>
```

```
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>The user's email.</returns>
    private static async Task<string> SubscribeToSnsTopic(string topicArn)
    {
        Console.WriteLine(new string('-', 80));


        string email = "";
        while (string.IsNullOrEmpty(email))
        {
            Console.WriteLine("Enter your email to subscribe to the Amazon SNS
topic:");
            email = Console.ReadLine()!;
        }

        var subscriptions = new List<string>();
        var paginatedSubscriptions =
_snsClient!.Paginators.ListSubscriptionsByTopic(
            new ListSubscriptionsByTopicRequest()
            {
                TopicArn = topicArn
            });

        // Get the entire list using the paginator.
        await foreach (var subscription in paginatedSubscriptions.Subscriptions)
        {
            subscriptions.Add(subscription.Endpoint);
        }

        if (subscriptions.Contains(email))
        {
            Console.WriteLine($"\tYour email is already subscribed.");
            Console.WriteLine(new string('-', 80));
            return email;
        }

        await _snsClient.SubscribeAsync(new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "email",
            Endpoint = email
        });
```

```
        Console.WriteLine($"Use the link in the email you received to confirm
your subscription, then press Enter to continue.");

        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
        return email;
    }

    /// <summary>
    /// Add a rule which triggers when a file is uploaded to an S3 bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role used by EventBridge.</param>
    /// <returns>Async task.</returns>
    private static async Task AddEventRule(string roleArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an EventBridge event that sends an email when
an Amazon S3 object is created.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await _eventBridgeWrapper.PutS3UploadRule(roleArn, eventRuleName,
testBucketName);
        Console.WriteLine($"\tAdded event rule {eventRuleName} for bucket
{testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an SNS target to the rule.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task AddSnsTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Adding a target to the rule to that sends an email
when the rule is triggered.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];
```

```
        var topicName = _configuration["topicName"];
        await _eventBridgeWrapper.AddSnsTargetToRule(eventRuleName, topicArn);
        Console.WriteLine($"\tAdded event rule {eventRuleName} with Amazon SNS
target {topicName} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the event rules on the default event bus.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListEventRules()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Current event rules:");

        var rules = await _eventBridgeWrapper.ListAllRulesForEventBus();
        rules.ForEach(r => Console.WriteLine($"\tRule: {r.Name} Description:
{r.Description} State: {r.State}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the event target to use a transform.
    /// </summary>
    /// <param name="topicArn">The SNS topic ARN target to update.</param>
    /// <returns>Async task.</returns>
    private static async Task UpdateSnsEventRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Let's update the event target with a transform.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await
_eventBridgeWrapper.UpdateS3UploadRuleTargetWithTransform(eventRuleName,
topicArn);
        Console.WriteLine($"\tUpdated event rule {eventRuleName} with Amazon SNS
target {topicArn} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Update the rule to use a custom event pattern.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UpdateToCustomRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Updating the event pattern to be triggered by a custom
event instead.");

        var eventRuleName = _configuration["eventRuleName"];

        await _eventBridgeWrapper.UpdateCustomEventPattern(eventRuleName);

        Console.WriteLine($"\tUpdated event rule {eventRuleName} to custom
pattern.");
        await
_eventBridgeWrapper.UpdateCustomRuleTargetWithTransform(eventRuleName,
            topicArn);

        Console.WriteLine($"\tUpdated event target {topicArn}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Send rule events for a custom rule using the user's email address.
    /// </summary>
    /// <param name="email">The email address to include.</param>
    /// <returns>Async task.</returns>
    private static async Task TriggerCustomRule(string email)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Sending an event to trigger the rule. This will
trigger a subscription email.");

        await _eventBridgeWrapper.PutCustomEmailEvent(email);

        Console.WriteLine($"\tEvents have been sent. Press Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// List all of the targets for a rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListTargets()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the targets for a particular rule.");

        var eventRuleName = _configuration["eventRuleName"];
        var targets = await
_eventBridgeWrapper.ListAllTargetsOnRule(eventRuleName);
        targets.ForEach(t => Console.WriteLine($"\tTarget: {t.Arn} Id: {t.Id}
Input: {t.Input}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the rules for a particular target.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task ListRulesForTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the rules for a particular target.");

        var rules = await _eventBridgeWrapper.ListAllRuleNamesByTarget(topicArn);
        rules.ForEach(r => Console.WriteLine($"\tRule: {r}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Enable or disable a particular rule.
    /// </summary>
    /// <param name="isEnabled">True to enable the rule, otherwise false.</param>
    /// <returns>Async task.</returns>
    private static async Task ChangeRuleState(bool isEnabled)
    {
        Console.WriteLine(new string('-', 80));
```

```
        var eventRuleName = _configuration["eventRuleName"];

        if (!isEnabled)
        {
            Console.WriteLine($"Disabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.DisableRuleByName(eventRuleName);
        }
        else
        {
            Console.WriteLine($"Enabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.EnableRuleByName(eventRuleName);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the current state of the rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetRuleState()
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        var state = await
_eventBridgeWrapper.GetRuleStateByRuleName(eventRuleName);
        Console.WriteLine($"Rule {eventRuleName} is in current state {state}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic to clean up.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        var eventRuleName = _configuration["eventRuleName"];
```

```
        if (GetYesNoResponse($"\tDelete all targets and event rule
{eventRuleName}? (y/n)"))
        {
            Console.WriteLine($"\tRemoving all targets from the event rule.");
            await _eventBridgeWrapper.RemoveAllTargetsFromRule(eventRuleName);

            Console.WriteLine($"\tDeleting event rule.");
            await _eventBridgeWrapper.DeleteRuleByName(eventRuleName);
        }

        var topicName = _configuration["topicName"];
        if (GetYesNoResponse($"\tDelete Amazon SNS subscription topic
{topicName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting topic.");
            await _snsClient!.DeleteTopicAsync(new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        }

        var bucketName = _configuration["testBucketName"];
        if (GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.S3Objects
                    .Select(o => new KeyVersion { Key = o.Key }).ToList()
            });
            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }
```

```
        var roleName = _configuration["roleName"];
        if (GetYesNoResponse($"\tDelete role {roleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policy and deleting role.");

            await _iamClient!.DetachRolePolicyAsync(new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
            });

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = roleName
            });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

創建一個包裝 EventBridge 操作的類。

```
/// <summary>
```

```
/// Wrapper for Amazon EventBridge operations.
/// </summary>
public class EventBridgeWrapper
{
    private readonly IAmazonEventBridge _amazonEventBridge;
    private readonly ILogger<EventBridgeWrapper> _logger;

    /// <summary>
    /// Constructor for the EventBridge wrapper.
    /// </summary>
    /// <param name="amazonEventBridge">The injected EventBridge client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public EventBridgeWrapper(IAmazonEventBridge amazonEventBridge,
 ILogger<EventBridgeWrapper> logger)

    {
        _amazonEventBridge = amazonEventBridge;
        _logger = logger;
    }

    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
 eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }

    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable a particular rule on an event bus.
    /// </summary
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
```

```
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
```

```
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
 _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                               "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });
```

```
        return response.RuleArn;
    }

    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
```

```
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Update a custom rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
 default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateCustomRuleTargetWithTransform(string
 ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputTemplate = "\"Notification: sample event was received.
\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
```

```
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }

    /// <summary>
    /// Update a rule to use a custom defined event pattern.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };
```

```
        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);
```

```
        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
 code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteRuleByName(string ruleName)
    {
        var response = await _amazonEventBridge.DeleteRuleAsync(
            new DeleteRuleRequest()
            {
                Name = ruleName
            });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - DeleteRule
  - DescribeRule
  - DisableRule

- EnableRule
- ListRuleNamesByTarget
- ListRules
- ListTargetsByRule
- PutEvents
- PutRule
- PutTargets

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
```

```
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException,
 IOException {
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name to create.
                    topicName - The name of the Amazon Simple Notification
 Service (Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
```

```
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM)
role to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the
user subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to
the specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email
when an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
```

```
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to
the S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a
custom pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn,
eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has
successfully completed.");
```

```
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient,
SnsClient snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String
bucketName, String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from
role " + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
```

```
        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
```

```
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();
```

```
        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
```

```
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   // Create and upload a file to an S3 bucket to trigger an event.
```

```
   public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
       // Create a unique file name.
       String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
       String fileName = "TextFile" + fileSuffix + ".txt";

       File myFile = new File(fileName);
       FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
       BufferedWriter bw = new BufferedWriter(fw);
       bw.write("This is a sample file for testing uploads.");
       bw.close();

       try {
           PutObjectRequest putOb = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(fileName)
                   .build();

           s3Client.putObject(putOb, RequestBody.fromFile(myFile));

       } catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();

       ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
       List<String> rules = response.ruleNames();
       for (String rule : rules) {
           System.out.println("The rule name is " + rule);
       }
   }

   public static void listTargets(EventBridgeClient eventBrClient, String
ruleName) {
```

```
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
```

```
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
"\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " +
rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
```

```
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();
```

```
            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String
bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
```

```
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();
```

```
            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets
 the user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon
 SNS) topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use
 with Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
```

```
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to
the topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
```

```
    println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName,
bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
```

```
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a
 subscription email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully
 completed.")
    println(DASHES)
}

suspend fun cleanupResources(topicArn: String?, eventRuleName: String?,
 bucketName: String?, roleName: String?) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
```

```
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest = DetachRolePolicyRequest {
        policyArn = policyArnVal
        roleName = roleNameVal
    }
    IamClient { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects = ListObjectsRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    }
                )
            }
```

```
        }

        val delOb = Delete {
            objects = toDelete
        }

        val dor = DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest = DeleteBucketRequest {
            bucket = bucketName
        }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest = DeleteRuleRequest {
        name = ruleName
    }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
```

```
    // First, get all targets that will be deleted.
    val request = ListTargetsByRuleRequest {
        rule = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest = RemoveTargetsRequest {
                    rule = eventRuleName
                    ids = listOf(myTarget.id.toString())
                }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json = "{" +
        "\"UserEmail\": \"" + email + "\"," +
        "\"Message\": \"This event was generated by example code.\"" +
        "\"UtcTime\": \"Now.\"" +
        "}"

    val entry = PutEventsRequestEntry {
        source = "ExampleSource"
        detail = json
        detailType = "ExampleType"
    }

    val eventsRequest = PutEventsRequest {
        this.entries = listOf(entry)
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

```
suspend fun updateCustomRuleTargetWithTransform(topicArn: String?, ruleName:
 String?) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb = InputTransformer {
        inputTemplate = "\"Notification: sample event was received.\""
    }

    val target = Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransformerOb
    }

    val targetsRequest = PutTargetsRequest {
        rule = ruleName
        targets = listOf(target)
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"]," +
        "\"detail-type\": [\"ExampleType\"]" +
        "}"
    val request = PutRuleRequest {
        name = ruleName
        description = "Custom test rule"
        eventPattern = customEventsPattern
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(topicArn: String?, ruleName: String?) {
```

```
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb = InputTransformer {
        inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
        inputPathsMap = myMap
    }
    val targetOb = Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransOb
    }

    val targetsRequest = PutTargetsRequest {
        rule = ruleName
        targets = listOf(targetOb)
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest = DescribeRuleRequest {
        name = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(eventRuleName: String, isEnabled: Boolean?) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest = DisableRuleRequest {
            name = eventRuleName
        }
```

```
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest = EnableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb = PutObjectRequest {
        bucket = bucketName
        key = fileName
        body = myFile.asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest = ListRuleNamesByTargetRequest {
        targetArn = topicArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
```

```
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest = ListTargetsByRuleRequest {
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(ruleName: String?, topicArn: String?, topicName:
 String, eventRuleName: String, bucketName: String) {
    val targetID = UUID.randomUUID().toString()
    val myTarget = Target {
        id = targetID
        arn = topicArn
    }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request = PutTargetsRequest {
        eventBusName = null
        targets = targetsOb
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
}
```

```
suspend fun subEmail(topicArnVal: String?, email: String?) {
    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\"," +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Resource\": \"*\"," +
        "\"Action\": \"sns:Publish\"" +
        "}]" +
        "}"

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}
```

```
suspend fun listRules() {
    val rulesRequest = ListRulesRequest {
        eventBusName = "default"
        limit = 10
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(roleArnVal: String?, bucketName: String, eventRuleName:
 String?) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest = PutRuleRequest {
        description = "Created by using the AWS SDK for Kotlin"
        name = eventRuleName
        eventPattern = pattern
        roleArn = roleArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
```

```
    val eventBridgeConfig = EventBridgeConfiguration {
    }

    val configuration = NotificationConfiguration {
        eventBridgeConfiguration = eventBridgeConfig
    }

    val configurationRequest = PutBucketNotificationConfigurationRequest {
        bucket = bucketName
        notificationConfiguration = configuration
        skipDestinationValidation = true
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }

        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
```

```
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(rolenameVal: String?, polJSON: String?): String? {
    val request = CreateRoleRequest {
        roleName = rolenameVal
        assumeRolePolicyDocument = polJSON
        description = "Created using the AWS SDK for Kotlin"
    }

    val rolePolicyRequest = AttachRolePolicyRequest {
        roleName = rolenameVal
        policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    }

    IamClient { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 的跨服務 EventBridge 範 AWS 例

下列範例應用程式使用 AWS SDK EventBridge 與其他 AWS 服務應用程式結合使用。每個範例都包含一個連結 GitHub，您可以在其中找到如何設定和執行應用程式的指示。

範例

- 使用排程事件來調用 Lambda 函數

使用排程事件來調用 Lambda 函數

下列程式碼範例說明如何建立 Amazon EventBridge 排程事件所叫用的 AWS Lambda 函數。

Java

適用於 Java 2.x 的 SDK

說明如何建立叫用 AWS Lambda 函數的 Amazon EventBridge 排程事件。設定 EventBridge 為在叫用 Lambda 函數時使用 cron 運算式來排程。在此範例中，您會使用 Lambda Java 執行期 API 建立 Lambda 函數。此範例會呼叫不同的 AWS 服務來執行特定使用案例。此範例示範如何建立應用程式，將行動裝置文字訊息傳送給員工，在他們的週年紀念日向他們道賀。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的開發套件

說明如何建立叫用 AWS Lambda 函數的 Amazon EventBridge 排程事件。設定 EventBridge 為在叫用 Lambda 函數時使用 cron 運算式來排程。在此範例中，您可以使用 Lambda JavaScript

執行階段 API 建立 Lambda 函數。此範例會呼叫不同的 AWS 服務來執行特定使用案例。此範例示範如何建立應用程式，將行動裝置文字訊息傳送給員工，在他們的週年紀念日向他們道賀。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例也可在 AWS SDK for JavaScript v3 開發人員指南中取得。

此範例中使用的服務

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

適用於 Python (Boto3) 的 SDK

此範例顯示如何將 AWS Lambda 函數註冊為已排程 Amazon EventBridge 事件的目標。Lambda 處理常式會將友善的訊息和完整事件資料寫入 Amazon CloudWatch 日誌，以供日後擷取。

- 部署 Lambda 函式。
- 建立 EventBridge 排程的事件，並使 Lambda 函數成為目標。
- 授予允許 EventBridge 叫用 Lambda 函數的權限。
- 列印記 CloudWatch 錄中的最新資料，以顯示排程呼叫的結果。
- 清理示範期間建立的所有資源。

此範例最佳檢視於 GitHub。有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- CloudWatch 日誌
- EventBridge
- Lambda

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 EventBridge 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Amazon EventBridge 安全

Amazon EventBridge 用 AWS Identity and Access Management 來控制對其他 AWS 服務和資源的訪問。如需 IAM 運作方式的概觀，請參閱《IAM 使用者指南》中的存取管理概觀。如需安全憑證的概觀，請參閱《Amazon Web Services 一般參考》中的 AWS 安全憑證。

主題

Amazon EventBridge 的資料保護

AWS 共同的責任模型適用於 Amazon EventBridge 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、AWS CLI 或 AWS 開發套件與 EventBridge 或其他 AWS 服務 協作時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

靜態加密

EventBridge 會加密其儲存的事件中繼資料和訊息資料。預設情況下，EventBridge 在 AWS 擁有的金鑰 下使用 256 位元進階加密標準 (AES-256) 來加密資料，這有助於保護您的資料免遭未經授權的存取。使用 AWS 擁有的金鑰來加密您的資料，則您的資料不收取額外費用。

傳輸中加密

在 EventBridge 和其他服務之間傳輸的所有資料都會使用 Transport Layer Security (TLS) 加密。

標籤類型政策

您可以在 Amazon EventBridge 中使用基於政策的標籤來控制對資源的存取。

例如，您可以限制包含鍵為 `environment`，值為 `production` 標籤的資源存取。下列範例政策禁止任何帶有此標籤的資源為已標記為 `environment/production` 的資源建立、刪除或修改標籤、規則或事件匯流排。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "events:PutRule",
                "events:DescribeRule",
                "events:DeleteRule",
                "events:CreateEventBus",
                "events:DescribeEventBus"
                "events:DeleteEventBus"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

如需標記的詳細資訊，請參閱以下內容：

- Amazon EventBridge 標籤
- 使用 IAM 標籤控制存取

Amazon EventBridge 和 AWS Identity and Access Management

若要存取 Amazon EventBridge，您需要AWS可用來驗證請求的登入資料。您的登入資料必須擁有許可以存取 AWS 資源，例如：從其他 AWS 資源擷取事件資料。以下各節提供如何使用 AWS Identity and Access Management(IAM) 的詳細資訊，以及透過控制可存取資源的人員 EventBridge 來協助保護資源的安全。

主題

身分驗證

您可以使用下列身分類型來存取 AWS：

- AWS 帳戶根使用者：在註冊 AWS 時，您會提供與您的帳戶相關聯的電子郵件地址和密碼。這些是您的根憑證，可完整存取您的所有 AWS 資源。

> **Important**
>
> 基於安全理由，建議您只在建立管理員時使用根登入資料，管理員也就是擁有您帳戶完整許可的 IAM 使用者。接著，您可以使用此管理員來建立其他使用者和角色，授予其有限的許可。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#create-iam-users中的IAM 最佳實務和建立 Admin (管理員) 使用者和群組。

- IAM 使用者 — IAM 使用者是您帳戶中具有特定許可的身分，例如，將事件資料傳送到中的目標的權限 EventBridge。您可以使用 IAM 登入憑證登入安全 AWS 網頁，例如： AWS Management Console、AWS 開發論壇或 AWS Support中心。

  除了登入憑證外，您還可以為每個使用者產生存取金鑰。以程式設計的方式存取 AWS 服務時，您可以使用這些金鑰加密簽署請求，方法是透過其中一個 SDK 或使用 AWS Command Line Interface (AWS CLI)。如果您不使用 AWS 工具，便必須使用簽章版本 4 (用來驗證送入 API 請求的協議) 自行簽署請求。如需有關驗證請求的詳細資訊，請參閱《Amazon Web Services 一般參考》中的 Signature 第 4 版簽署程序。
- IAM 角色：IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。它類似 IAM 使用者，但未與特定的人員建立關聯。您可利用 IAM 角色取得臨時存取金鑰，以存取 AWS 服務和資源。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：非建立使用者，而是使用來自 AWS Directory Service、您的企業使用者目錄或 Web 身分提供者 (IdP) 的使用者身分。這些稱為 聯合身分使用者。利用身分提供者來請求存取時，AWS 會指派角色給聯合身分使用者。如需有關聯合身分使用者的詳細資訊，請參閱 IAM 使用者指南中的聯合身分使用者和角色。
  - 跨帳户存取權：您可以使用帳戶中的 IAM 角色，授予另一個帳戶許可來存取您帳戶中的資源。如需範例，請參閱《IAM 使用者指南》中的教學課程：使用 IAM 角色將存取權委派給不同的 AWS 帳戶。
  - AWS 服務存取：您可以使用帳戶中的 IAM 角色，授予 AWS 服務許可來存取您帳戶中的資源。例如，您可以建立角色，以允許 Amazon Redshift 將存放於 Amazon S3 儲存貯體中的資料載入到 Amazon Redshift 叢集。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務。
  - 在 Amazon EC2 上執行的應用程式 — 對於需要存取權的 Amazon EC2 應用程式 EventBridge，您可以在 EC2 執行個體中存放存取金鑰，或使用 IAM 角色來管理臨時登入資料。若要將 AWS 角色指派給 EC2 執行個體，您需建立執行個體描述檔，將其附加到執行個體。執行個體描述檔包含該角色，並且會為在 EC2 執行個體上執行的程式提供臨時憑證。如需詳細資訊，請參閱 IAM 使用者指南https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html中的為 Amazon EC2 上的應用程式使用角色。

存取控制

若要建立或存取 EventBridge 資源，您需要有效的認證和權限。例如，若要叫用 AWS Lambda、Amazon Simple Notification Service (Amazon SNS) 和 Amazon Simple Queue Service (Amazon SQS) 目標，您必須擁有對這些服務的許可。

管理對您 Amazon EventBridge 資源的存取許可

您可以使用以身分識別為基礎或以資源為基礎的政策來管理 EventBridge 資源 (例如規則或事件) 的存取權。

EventBridge 資源

EventBridge 資源和子資源各與唯一的 Amazon Resource Name (ARN) 相關聯。您可以在 EventBridge 接中使用 ARN 來建立事件模式。如需 ARN 的詳細資訊，請參閱 Amazon Web Services 一般參考 中的 Amazon Resource Name (ARN) 與 AWS 服務命名空間。

如需 EventBridge 為使用資源所提供的作業清單，請參閱 Amazon EventBridge 許可參考。

> **Note**
>
> 大多數的 AWS 服務會將 ARN 中的冒號 (:) 或正斜線 (/) 視為相同字元。但是，EventBridge 會在事件模式和規則中使用完全相符。在建立事件模式時，請務必使用正確的 ARN 字元，使這些字元符合您要比對事件中的 ARN 語法。

下表顯示 EventBridge 的資源。

資源類型	ARN 格式
存檔	`arn:aws:events:` *`region`*`:`*`account`*`:archive/` *`archive-name`*
重新播放	`arn:aws:events:` *`region`*`:`*`account`*`:replay/`*`replay-name`*
規則	`arn:aws:events:` *`region`*`:`*`account`*`:rule/[event-bus-name]/`*`rule-name`*
事件匯流排	`arn:aws:events:` *`region`*`:`*`account`*`:event-bus/` *`event-bus-name`*
所有 EventBridge 資源	`arn:aws:events:*`

資源類型	ARN 格式
在指定區域中，指定之帳戶擁有的所有 EventBridge 資源	`arn:aws:events: region:account:*`

以下範例展示如何在您的陳述式中使用其 ARN 指定特定的規則 (*myRule*)。

```
"Resource": "arn:aws:events:us-east-1:123456789012:rule/myRule"
```

若要透過使用星號 (*) 萬用字元指定所有屬於特定帳戶的規則，如下所示。

```
"Resource": "arn:aws:events:us-east-1:123456789012:rule/*"
```

若要指定所有資源，或如果特定的 API 動作不支援 ARN，請在 Resource 元素中使用星號 (*) 萬用字元，如下所示。

```
"Resource": "*"
```

若要在單一陳述式中指定多項資源或 PutTargets，請用逗號分隔其 ARN，如下所示。

```
"Resource": ["arn1", "arn2"]
```

資源擁有權

帳戶擁有資源內的資源，無論資源的建立者是誰。資源擁有者就是驗證建立資源請求之 主體實體 (根使用者、IAM 使用者或 IAM 角色) 的 AWS 帳戶。下列範例說明其如何運作：

- 如果您使用帳戶的根使用者憑證來建立規則，則您的帳戶即為 EventBridge 資源的擁有者。
- 如果您在帳戶中建立使用者，並將建立 EventBridge 資源的許可授予該使用者，則該使用者可以建立 EventBridge 資源。不過，您的帳戶 (也是該使用者所屬的帳戶) 擁有該 EventBridge 資源。
- 如果您在您的帳戶中，建立了擁有可建立 EventBridge 資源之許可的 IAM 角色，則任何可能擔任該角色的人都能建立 EventBridge 資源。您的帳戶 (即該角色所屬帳戶) 擁有這些 EventBridge 資源。

管理資源存取

許可政策描述誰可以存取哪些資源。下一節說明可用來建立許可政策的選項。

> **Note**
>
> 本節著重討論如何在 EventBridge 的環境中使用 IAM，它不提供 IAM 服務的詳細資訊。如需完整的 IAM 文件，請參閱《IAM 使用者指南》中的什麼是 IAM？。如需有關 IAM 政策語法和說明的資訊，請參閱《IAM 使用者指南》中的 IAM 政策參考。

連接到 IAM 身分的政策稱為身分類型政策 (IAM 政策)，而連接到資源的政策參考資源類型政策。在 EventBridge 中，您可以合併使用身分類型 (IAM 政策) 和資源類型政策。

主題

- 身分類型政策 (IAM 政策)
- 以資源爲基礎的政策 (IAM 政策)

身分類型政策 (IAM 政策)

您可以將政策連接到 IAM 身分。例如，您可以執行下列動作：

- 將許可政策連接至您帳戶中的使用者或群組：若要授予使用者在 Amazon CloudWatch 主控台檢視日誌的許可，您可以將許可政策連接至使用者，或使用者所屬的群組。
- 將許可政策連接至角色 (授予跨帳戶許可)：您可以將身分識別型許可政策連接至 IAM 角色，藉此授予跨帳戶許可。例如，帳戶 A 的管理員可以建立角色，將跨帳戶許可授予另一個 B 帳戶或某個 AWS 服務，如下所示：
  1. 帳戶 A 管理員建立 IAM 角色，並將許可政策連接到可授與帳戶 A 中資源許可的角色。
  2. 帳戶 A 管理員會將信任政策連接至將帳戶 B 識別為可擔任角色之主體的角色。
  3. 帳戶 B 管理員接著可以將擔任該角色的許可委派給帳戶 B 中的任何使用者。這樣可讓帳戶 B 的使用者建立或存取帳戶 A 的資源。信任政策中的主體也可以是 AWS 服務主體，將擔任該角色的許可授予 AWS 服務。

  如需使用 IAM 來委派許可的詳細資訊，請參閱《IAM 使用者指南》中的存取管理。

您可以建立特定 IAM 政策，限制您帳戶中的使用者有權存取的呼叫和資源，然後將那些政策連接到使用者。如需有關如何建立 IAM 角色，以及探索範例 IAM 政策陳述式的詳細資訊，請參閱 管理對您 Amazon EventBridge 資源的存取許可。

以資源爲基礎的政策 (IAM 政策)

在 EventBridge 中執行規則時，會調用與該規則關聯的所有目標，也就是調用 AWS Lambda 函數、發佈到 Amazon SNS 主題，或將事件轉送至 Amazon Kinesis 串流。若要能夠根據您所擁有的資源進行 API 呼叫，EventBridge 需要適當的許可。對於 Lambda, Amazon SNS 和 Amazon SQS 資源，EventBridge 會使用基於資源的政策。對於 Kinesis 串流，EventBridge 使用 IAM 角色。

如需有關如何建立 IAM 角色，以及探索適用於 EventBridge 的以資源為基礎的範例政策陳述式的詳細資訊，請參閱 將以資源為基礎的政策用於 Amazon EventBridge。

指定政策元素：動作、效果和主體

對於每項 EventBridge 資源，EventBridge 定義一組 API 操作。EventBridge 定義一組您可在政策中指定的動作，以授予這些 API 操作的許可。某些 API 操作可能需要多個動作的許可來執行 API 操作。如需資源與 API 操作的詳細資訊，請參閱 EventBridge 資源 與 Amazon EventBridge 許可參考.

以下是基本的政策元素：

- 資源：使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。如需更多詳細資訊，請參閱 EventBridge 資源。
- 動作：使用動作關鍵字來識別您要允許或拒絕的資源操作。例如，`events:Describe` 許可允許使用者執行 `Describe` 操作。
- 效果：指定允許或拒絕。如果您未明確授予存取 (允許) 資源，則隱含地拒絕存取。您也可以明確拒絕資源的存取權，這樣做可以確保使用者無法存取資源，即使另有其他政策授與存取。
- 主體：在以身分為基礎的政策 (IAM 政策) 中，政策所連接的使用者就是隱含主體。對於資源類型政策，您可以指定想要收到許可的使用者、帳戶、服務或其他實體 (僅適用於資源類型政策)。

如需 IAM 政策語法和說明的詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策參考。

如需 EventBridge API 動作表和它們所套用的資源，請參閱 Amazon EventBridge 許可參考。

在政策中指定條件

當您授予許可時，可以使用存取政策語言來指定政策應該何時生效的條件。例如，建議只在特定日期之後套用政策。如需使用政策語言指定條件的詳細資訊，請參閱IAM 使用者指南中的條件。

欲定義條件，您可以使用條件鍵。有您可以視需要使用的 AWS 條件金鑰和 EventBridge 特定鍵。如需全 AWS 鍵的完整清單，請參閱《IAM 使用者指南》中的可用的條件索引鍵。如需 EventBridge 專屬索引鍵的完整清單，請參閱 使用 IAM 政策條件進行精細定義存取控制。

將以身分為基礎的政策 (IAM 政策) 用於 Amazon EventBridge

以身分為基礎的政策是指您連接到 IAM 身分的許可政策。

主題

適用於 EventBridge 的 AWS 受管政策

AWS 透過提供獨立的 IAM 政策來解決許多常用案例，這些政策由 AWS 所建立與管理。受管 或預定義政策會針對常用案例授予必要的許可，因此您無須調查需要哪些許可。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。

以下 AWS 受管政策 (您可以將它們連接到您帳戶中的使用者) 專屬於 EventBridge：

- AmazonEventBridgeFullAccess：授予對 EventBridge 的完整存取權，包括 EventBridge 管道、EventBridge 結構和 EventBridge 排程器。
- AmazonEventBridgeReadOnlyAccess：授予對 EventBridge 的唯讀存取權，包括 EventBridge 管道、EventBridge 結構和 EventBridge 排程器。

AmazonEventBridgeFullAccess 政策

AmazonEventBridgeFullAccess 政策授予使用所有 EventBridge 動作的權限以及下列權限：

- `iam:CreateServiceLinkedRole`：EventBridge 需要此權限才能在您的帳戶中針對 API 目的地建立服務角色。此權限僅授予 IAM 服務許可，以便在您的帳戶中專門針對 API 目的地建立角色。
- `iam:PassRole`：EventBridge 需要此權限才能將呼叫角色傳遞給 EventBridge，以呼叫規則的目標。
- 密碼管理員權限：當您使用連線資源來提供認證以授權 API 目的地時，EventBridge 需要這些權限來管理您帳戶中的密碼。

以下 JSON 顯示了 AmazonEventBridgeFullAccess 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EventBridgeActions",
            "Effect": "Allow",
            "Action": [
                "events:*",
                "schemas:*",
                "scheduler:*",
                "pipes:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMCreateServiceLinkedRoleForApiDestinations",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
AmazonEventBridgeApiDestinationsServiceRolePolicy",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "apidestinations.events.amazonaws.com"
                }
            }
        },
        {
            "Sid": "IAMCreateServiceLinkedRoleForAmazonEventBridgeSchemas",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/schemas.amazonaws.com/
AWSServiceRoleForSchemas",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "schemas.amazonaws.com"
                }
            }
        },
        {
            "Sid": "SecretsManagerAccessForApiDestinations",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:CreateSecret",
```

```
                "secretsmanager:UpdateSecret",
                "secretsmanager:DeleteSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:events!*"
        },
        {
            "Sid": "IAMPassRoleAccessForEventBridge",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "events.amazonaws.com"
                }
            }
        },
        {
            "Sid": "IAMPassRoleAccessForScheduler",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
        },
        {
            "Sid": "IAMPassRoleAccessForPipes",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "pipes.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> 本節中的資訊也適用於 `CloudWatchEventsFullAccess` 策略。但是，我們強烈建議您使用 Amazon EventBridge 代替 Amazon CloudWatch Events。

AmazonEventBridgeReadOnlyAccess 政策

AmazonEventBridgeReadOnlyAccess 政策授予使用所有讀取 EventBridge 動作的權限。

以下 JSON 顯示了 AmazonEventBridgeReadOnlyAccess 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:DescribeRule",
                "events:DescribeEventBus",
                "events:DescribeEventSource",
                "events:ListEventBuses",
                "events:ListEventSources",
                "events:ListRuleNamesByTarget",
                "events:ListRules",
                "events:ListTargetsByRule",
                "events:TestEventPattern",
                "events:DescribeArchive",
                "events:ListArchives",
                "events:DescribeReplay",
                "events:ListReplays",
                "events:DescribeConnection",
                "events:ListConnections",
                "events:DescribeApiDestination",
                "events:ListApiDestinations",
                "events:DescribeEndpoint",
                "events:ListEndpoints",
                "schemas:DescribeCodeBinding",
                "schemas:DescribeDiscoverer",
                "schemas:DescribeRegistry",
                "schemas:DescribeSchema",
                "schemas:ExportSchema",
                "schemas:GetCodeBindingSource",
```

```
                "schemas:GetDiscoveredSchema",
                "schemas:GetResourcePolicy",
                "schemas:ListDiscoverers",
                "schemas:ListRegistries",
                "schemas:ListSchemas",
                "schemas:ListSchemaVersions",

                "schemas:ListTagsForResource",
                "schemas:SearchSchemas",
                "scheduler:GetSchedule",
                "scheduler:GetScheduleGroup",
                "scheduler:ListSchedules",
                "scheduler:ListScheduleGroups",
                "scheduler:ListTagsForResource",
                "pipes:DescribePipe",
                "pipes:ListPipes",
                "pipes:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 本節中的資訊也適用於 `CloudWatchEventsReadOnlyAccess` 政策。但是，我們強烈建議您使用 Amazon EventBridge 代替 Amazon CloudWatch Events。

適用於 EventBridge 結構描述的受管政策

結構描述 會定義傳送至 EventBridge 的事件結構。EventBridge 會為 AWS 服務所產生的所有事件提供結構描述。下列是適用於 EventBridge 結構描述的 AWS 受管政策：

- AmazonEventBridgeSchemasServiceRolePolicy
- AmazonEventBridgeSchemasFullAccess
- AmazonEventBridgeSchemasReadOnlyAccess

適用於 EventBridge 排程器的受管政策

Amazon EventBridge 排程器是無伺服器排程器，可讓您從單一受管的中央服務建立、執行並管理任務。如需 AWS 排程器特定的受管政策, 請參閱《EventBridge 排程器用戶指南》 中 EventBridge 排程器的 AWS 受管政策。

適用於 EventBridge 管道的受管政策

Amazon EventBridge 管道將事件來源連接到目標。開發事件驅動的架構時，管道可減少對專業知識和整合程式碼的需求。這有助於確保您公司應用程式的一致性。下列是適用於 EventBridge 管道的 AWS 受管政策：

- AmazonEventBridgePipesFullAccess

  提供對 Amazon EventBridge 管道的完整存取。

  > **Note**
  >
  > 此政策提供 `iam:PassRole`：EventBridge 管道需要此權限，才能將呼叫角色傳遞給 EventBridge 以建立及啟動管道。

- AmazonEventBridgePipesReadOnlyAccess

  提供對 Amazon EventBridge 管道的唯讀存取權限。

- AmazonEventBridgePipesOperatorAccess

  提供對 Amazon EventBridge 管道的唯讀和運算子 (即：停止和開始執行管道的功能) 存取權。

傳送事件的 IAM 角色

若要將事件轉送至目標，EventBridge 需要 IAM 角色。

若要建立 IAM 角色以將事件傳送至 EventBridge

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 若要建立 IAM 角色，依照《IAM 使用者指南》 中的 建立角色以委派許可給 AWS 服務的步驟。按照步驟執行時，請執行下列操作：

   - 在角色名稱中，使用您帳戶中的唯一名稱。

- 在選擇角色類型中，先選擇 AWS 服務角色，然後選擇 Amazon EventBridge。此操作會授予 EventBridge 擔任該角色的許可。
- 在附加政策中，選擇 AmazonEventBridgeFullAccess。

您也可以建立自己的自訂 IAM 政策，以允許 EventBridge 動作與資源的許可。您可以將這些自訂政策連接至需要這些許可的 IAM 使用者或群組。如需關於 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》 中的 IAM 政策概觀。如需關於管理和建立自訂 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》中的管理 IAM 政策。

EventBridge 使用 IAM 角色存取目標所需的權限

EventBridge 目標通常需要將權限授與 EventBridge 呼叫目標的 IAM 角色。以下是各種 AWS 服務和目標的一些範例。對於其他人，請使用 EventBridge 主控台建立規則並建立新的角色。該角色將使用預先設定完善範圍的權限的政策來建立。

Amazon SQS、Amazon SNS、Lambda、CloudWatch Logs 和 EventBridge 匯流排目標不使用角色，而且必須透過資源政策授予 EventBridge 的許可。API 閘道目標可以使用資源政策或 IAM 角色。

如果目標是 API 目的地，您指定的角色必須包含下列政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Action": [ "events:InvokeApiDestination" ],
            "Resource": [ "arn:aws:events:::api-destination/*" ]
        }
     ]
}
```

如果目標是 Kinesis 串流，用於傳送事件資料到該目標的角色必須包含下列政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

如果目標是 Systems Manager Run Command，而且您為該命令指定一或多個 `InstanceIds` 值，則您指定的角色必須包含下列政策。

```
{
     "Version": "2012-10-17",
     "Statement": [
          {
               "Action": "ssm:SendCommand",
               "Effect": "Allow",
               "Resource": [
                    "arn:aws:ec2:region:accountId:instance/instanceIds",
                    "arn:aws:ssm:region:*:document/documentName"
               ]
          }
     ]
}
```

如果目標是 Systems Manager 運行命令，而且您為該命令指定一或多個標籤，則您指定的角色必須包含下列政策：

```
{
     "Version": "2012-10-17",
     "Statement": [
          {
               "Action": "ssm:SendCommand",
               "Effect": "Allow",
               "Resource": [
                    "arn:aws:ec2:region:accountId:instance/*"
               ],
               "Condition": {
                   "StringEquals": {
                       "ec2:ResourceTag/*": [
                           "[[tagValues]]"
                       ]
                   }
               }
```

```
            },
            {
                 "Action": "ssm:SendCommand",
                 "Effect": "Allow",
                 "Resource": [
                      "arn:aws:ssm:region:*:document/documentName"
                 ]
            }
     ]
}
```

如果目標是 AWS Step Functions 狀態機器，您指定的角色必須包含下列政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Action": [ "states:StartExecution" ],
            "Resource": [ "arn:aws:states:*:*:stateMachine:*" ]
        }
     ]
}
```

如果目標是 Amazon ECS 任務，您指定的角色必須包含下列政策。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "ecs:RunTask"
        ],
        "Resource": [
            "arn:aws:ecs:*:account-id:task-definition/task-definition-name"
        ],
        "Condition": {
            "ArnLike": {
                "ecs:cluster": "arn:aws:ecs:*:account-id:cluster/cluster-name"
            }
        }
    },
    {
```

```
        "Effect": "Allow",
        "Action":"iam:PassRole",
        "Resource": [
            "*"
        ],
        "Condition": {
            "StringLike": {
                "iam:PassedToService": "ecs-tasks.amazonaws.com"
            }
        }
    }]
}
```

以下政策允許 EventBridge 中的內建目標代表您執行 Amazon EC2 動作。您需要使用 AWS Management Console 建立規則 (使用內建目標)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TargetInvocationAccess",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:RebootInstances",
                "ec2:StopInstances",
                "ec2:TerminateInstances",
                "ec2:CreateSnapshot"
            ],
            "Resource": "*"
        }
    ]
}
```

以下政策允許 EventBridge 將事件轉送至您帳戶中的 Kinesis 串流。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KinesisAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "kinesis:PutRecord"
            ],
            "Resource": "*"
        }
    ]
}
```

客戶管理政策範例：使用標記來控制規則的存取

以下範例顯示授予 EventBridge 動作權限的使用者政策。當您使用 EventBridge API、AWS SDK 或 AWS CLI 時，該政策適用。

您可以授予使用者存取特定的 EventBridge 規則，同時防止他們存取其他規則。為此，您需標記這兩個規則，並使用那些帶有標籤的 IAM 政策。如需關於標記 EventBridge 資源的詳細資訊，請參閱 Amazon EventBridge 標籤。

您可以授予一個 IAM 政策給使用者，僅允許存取具有特定標籤的規則。您可以使用該特定標記來標記規則，以選擇要授與存取權的規則。例如，以下政策會授予標籤鍵 `Stack` 值為 `Prod` 規則的存取。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "events:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Stack": "Prod"
                }
            }
        }
    ]
}
```

如需有關使用 IAM 政策陳述式的詳細資訊，請參閱《IAM 使用者指南》中的使用政策控制存取。

Amazon EventBridge 的 AWS 受管政策更新

檢視自該服務開始追蹤 EventBridge 的 AWS 受管政策變更以來的詳細更新資訊。如需關於此頁面變更的自動提醒，請訂閱 EventBridge 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AmazonEventBridgePipesFullAccess：新增了新政策	EventBridge 針對使用 EventBridge 管道的完整權限新增了受管政策。	2022 年 12 月 1 日
AmazonEventBridgePipesReadOnlyAccess：新增了新政策	EventBridge 為檢視 EventBridge 管道資訊資源的權限新增了受管政策。	2022 年 12 月 1 日
AmazonEventBridgePipesOperatorAccess：新增了新政策	EventBridge 新增了受管政策，以用於檢視 EventBridge 管道資訊的權限，以及啟動和停止執行中的管道。	2022 年 12 月 1 日
AmazonEventBridgeFullAccess：更新至現有政策	EventBridge 已更新政策，以納入使用 EventBridge 管道功能所需的權限。	2022 年 12 月 1 日
AmazonEventBridgeReadOnlyAccess：更新至現有政策	EventBridge 新增了檢視 EventBridge 管道資訊資源所需的權限。 已新增下列動作： • `pipes:DescribePipe` • `pipes:ListPipes` • `pipes:ListTagsForResource`	2022 年 12 月 1 日
CloudWatchEventsReadOnlyAccess：更新至現有政策	更新以匹配 AmazonEventBridgeReadOnlyAccess。	2022 年 12 月 1 日
CloudWatchEventsFullAccess：更新至現有政策	更新以匹配 AmazonEventBridgeFullAccess。	2022 年 12 月 1 日

變更	描述	日期
AmazonEventBridgeFullAccess：更新至現有政策	EventBridge 已更新政策，以納入使用結構描述和排程器功能所需的權限。 已新增下列許可： • EventBridge 結構描述登錄檔動作 • EventBridge 排程器動作 • EventBridge 結構描述登錄檔的 `iam:CreateServiceLinkedRole` 權限 • EventBridge 排程器的 `iam:PassRole` 權限	2022 年 11 月 10 日

變更	描述	日期
AmazonEventBridgeReadOnlyAccess：更新至現有政策	EventBridge 新增了檢視結構描述和排程器資訊資源所需的權限。 已新增下列動作： • schemas:DescribeCodeBinding • schemas:DescribeDiscoverer • schemas:DescribeRegistry • schemas:DescribeSchema • schemas:ExportSchema • schemas:GetCodeBindingSource • schemas:GetDiscoveredSchema • schemas:GetResourcePolicy • schemas:ListDiscoverers • schemas:ListRegistries • schemas:ListSchemas • schemas:ListSchemaVersions • schemas:ListTagsForResource	2022 年 11 月 10 日

變更	描述	日期
	• schemas:SearchSchemas • scheduler:GetSchedule • scheduler:GetScheduleGroup • scheduler:ListSchedules • scheduler:ListScheduleGroups • scheduler:ListTagsForResource	
AmazonEventBridgeReadOnlyAccess：更新至現有政策	EventBridge 已新增檢視端點資訊所需的權限。 已新增下列動作： • events:ListEndpoints • events:DescribeEndpoint	2022 年 4 月 7 日

變更	描述	日期
AmazonEventBridgeReadOnlyAccess：更新至現有政策	EventBridge 新增了檢視連線和 API 目標資訊所需的權限。 已新增下列動作： • events:DescribeConnection • events:ListConnections • events:DescribeApiDestination • events:ListApiDestinations	2021 年 3 月 4 日
AmazonEventBridgeFullAccess：更新至現有政策	EventBridge 更新了政策，以包含使用 API 目的地所需的 iam:CreateServiceLinkedRole 和 AWS Secrets Manager 權限。 已新增下列動作： • secretsmanager:CreateSecret • secretsmanager:UpdateSecret • secretsmanager:DeleteSecret • secretsmanager:GetSecretValue • secretsmanager:PutSecretValue	2021 年 3 月 4 日

變更	描述	日期
EventBridge 已開始追蹤變更	EventBridge 已開始追蹤其 AWS 受管政策的變更。	2021 年 3 月 4 日

將以資源為基礎的政策用於 Amazon EventBridge

EventBridge 中，規則觸發時，與該規則關聯的所有目標都會被叫用。規則可以叫用 AWS Lambda 函數，發佈到 Amazon SNS 主題，或將事件轉送到 Kinesis 串流。為了根據您所擁有的資源進行 API 呼叫，EventBridge 需要適當的許可。對於 Lambda、Amazon SNS、Amazon SQS 和 Amazon CloudWatch Logs 資源，EventBridge 使用以資源為基礎的政策。對於 Kinesis 串流，EventBridge 會使用 以身分為基礎的 政策。

您可以使用 AWS CLI 將權限新增至目標。如需如何安裝和設定 AWS CLI 的資訊，請參閱《AWS Command Line Interface 使用者指南》中的使用 AWS Command Line Interface 開始設定。

主題

- Amazon API 閘道許可
- CloudWatch Logs 許可
- AWS Lambda 許可
- Amazon SNS 許可
- Amazon SQS 許可
- EventBridge 管道詳細資訊

Amazon API 閘道許可

若要使用 EventBridge 規則叫用 Amazon API 閘道端點，請在 API 閘道端點的政策中新增下列權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Service": "events.amazonaws.com"
           },
           "Action": "execute-api:Invoke",
           "Condition": {
               "ArnEquals": {
                   "aws:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"
               }
           },
           "Resource": [
               "execute-api:/stage/GET/api"
```

```
            ]
        }
    ]
}
```

CloudWatch Logs 許可

當 CloudWatch Logs 是規則的目標時，EventBridge 會建立日誌串流，且 CloudWatch Logs 會將事件中的文字存為日誌項目。若要讓 EventBridge 建立日誌串流並記錄事件，CloudWatch Logs 必須包含以資源為基礎的政策，此政策可讓 EventBridge 寫入 CloudWatch Logs。

如果您使用 AWS Management Console 將 CloudWatch Logs 新增為規則的目標，系統就會自動建立以資源為基礎的政策。如果您使用 AWS CLI 來新增目標，但此政策不存在，那麼您必須建立此政策。

下列範例可讓 EventBridge 寫入名稱開頭為 `/aws/events/` 的所有日誌群組。如果您將不同的命名政策用於這些日誌類型，請據此調整範例。

```
{
    "Statement": [
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": ["events.amazonaws.com", "delivery.logs.amazonaws.com"]
            },
            "Resource": "arn:aws:logs:region:account:log-group:/aws/events/*:*",
            "Sid": "TrustEventsToStoreLogEvent"
        }
    ],
    "Version": "2012-10-17"
}
```

如需詳細資訊，請參閱《Amazon CloudWatch API 參考》 中的 PutResourcePolicy。

AWS Lambda 許可

若要使用 EventBridge 規則叫用您的 AWS Lambda 函數，請將以下許可新增至您的 Lambda 函數政策。

```
{
  "Effect": "Allow",
  "Action": "lambda:InvokeFunction",
  "Resource": "arn:aws:lambda:region:account-id:function:function-name",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Condition": {
    "ArnLike": {
      "AWS:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"
    }
  },
  "Sid": "InvokeLambdaFunction"
}
```

若要新增上述允許 EventBridge 使用 AWS CLI 叫用 Lambda 函數的上述權限

- 在命令提示中，輸入以下命令：

```
aws lambda add-permission --statement-id "InvokeLambdaFunction" \
--action "lambda:InvokeFunction" \
--principal "events.amazonaws.com" \
--function-name "arn:aws:lambda:region:account-id:function:function-name" \
--source-arn "arn:aws:events:region:account-id:rule/rule-name"
```

如需有關設定許可使 EventBridge 能夠叫用 Lambda 函數的詳細資訊，請參閱《AWS Lambda 開發人員指南》 中的 AddPermission 和 使用 Lambda 與排程事件。

Amazon SNS 許可

若要允許 EventBridge 發佈到 Amazon SNS 主題，請使用 `aws sns get-topic-attributes` 和 `aws sns set-topic-attributes` 命令。

> **Note**
>
> 您無法將 Amazon SNS 主題政策中的 `Condition` 區塊用於 EventBridge。

若要新增許可使 EventBridge 發佈 SNS 主題

1. 若要列出 SNS 主題的屬性，請使用下列命令。

```
aws sns get-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name"
```

以下範例顯示新 SNS 主題的結果。

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "0",
        "DisplayName": "",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy\":
{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,\"numMaxDelayRetries
\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,\"backoffFunction\":\"linear
\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "account-id",
        "Policy": "{\"Version\":\"2012-10-17\",\"Id\":\"__default_policy_ID\",
\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":\"Allow\",\"Principal
\":{\"AWS\":\"*\"},\"Action\":[\"SNS:GetTopicAttributes\",\"SNS:SetTopicAttributes
\",\"SNS:AddPermission\",\"SNS:RemovePermission\",\"SNS:DeleteTopic\",
\"SNS:Subscribe\",\"SNS:ListSubscriptionsByTopic\",\"SNS:Publish\"],\"Resource
\":\"arn:aws:sns:region:account-id:topic-name\",\"Condition\":{\"StringEquals\":
{\"AWS:SourceOwner\":\"account-id\"}}}]}",
        "TopicArn": "arn:aws:sns:region:account-id:topic-name",
        "SubscriptionsPending": "0"
    }
}
```

2. 使用JSON 到字串轉換器將以下語句轉換為字符串。

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Action": "sns:Publish",
  "Resource": "arn:aws:sns:region:account-id:topic-name"
}
```

下列範例為將語句轉換為字串後的狀態。

```
{\"Sid\":\"PublishEventsToMyTopic\",\"Effect\":\"Allow\",\"Principal\":
{\"Service\":\"events.amazonaws.com\"},\"Action\":\"sns:Publish\",\"Resource\":
\"arn:aws:sns:region:account-id:topic-name\"}
```

3. 將您在上一個步驟中建立的字串新增至 "Policy" 屬性內的 "Statement" 集合中。
4. 使用 aws sns set-topic-attributes 命令設定新政策。

```
aws sns set-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name"
 \
--attribute-name Policy \
--attribute-value "{\"Version\":\"2012-10-17\",\"Id\":\"__default_policy_ID\",
\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":\"Allow\",\"Principal
\":{\"AWS\":\"*\"},\"Action\":[\"SNS:GetTopicAttributes\",\"SNS:SetTopicAttributes
\",\"SNS:AddPermission\",\"SNS:RemovePermission\",\"SNS:DeleteTopic\",
\"SNS:Subscribe\",\"SNS:ListSubscriptionsByTopic\",\"SNS:Publish\"],\"Resource
\":\"arn:aws:sns:region:account-id:topic-name\",\"Condition\":{\"StringEquals
\":{\"AWS:SourceOwner\":\"account-id\"}}}, {\"Sid\":\"PublishEventsToMyTopic\",
\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},\"Action
\":\"sns:Publish\",\"Resource\":\"arn:aws:sns:region:account-id:topic-name\"}]}"
```

如需詳細資訊，請前往 Amazon Simple Notification Service API 參考中的 SetTopicAttributes。

Amazon SQS 許可

若要允許 EventBridge 規則呼叫 Amazon SQS 佇列，請使用 aws sqs get-queue-attributes 和 aws sqs set-queue-attributes 命令。

如果 SQS 佇列的政策為空，您需要先建立政策，然後才能在其中新增權限陳述式。新的 SQS 佇列具有空白政策。

如果 SQS 佇列已有政策，您需要複製原始政策，並將它與新陳述式結合，才能在其中新增權限陳述式。

新增許可使 EventBridge 規則能夠呼叫 SQS 佇列

1. 列出 SQS 佇列屬性。在命令提示中，輸入以下命令：

```
aws sqs get-queue-attributes \
```

```
--queue-url https://sqs.region.amazonaws.com/account-id/queue-name \
--attribute-names Policy
```

2. 添加以下陳述式。

```
{
      "Sid": "AWSEvents_custom-eventbus-ack-sqs-rule_dlq_sqs-rule-target",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:region:account-id:queue-name",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:events:region:account-id:rule/bus-name/rule-
name"
        }
      }
    }
```

3. 使用 JSON 到字串轉換器將前面的語句轉換為字串。下列為將政策轉換為字串後的狀態。

```
{\"Sid\": \"EventsToMyQueue\", \"Effect\": \"Allow\", \"Principal\": {\"Service
\": \"events.amazonaws.com\"}, \"Action\": \"sqs:SendMessage\", \"Resource\":
 \"arn:aws:sqs:region:account-id:queue-name\", \"Condition\": {\"ArnEquals\":
 {\"aws:SourceArn\": \"arn:aws:events:region:account-id:rule/rule-name\"}}
```

4. 建立稱為 set-queue-attributes.json 的檔案，其中具有以下內容。

```
{
    "Policy": "{\"Version\":\"2012-10-17\",\"Id\":\"arn:aws:sqs:region:account-
id:queue-name/SQSDefaultPolicy\",\"Statement\":[{\"Sid\": \"EventsToMyQueue\",
 \"Effect\": \"Allow\", \"Principal\": {\"Service\": \"events.amazonaws.com\"},
 \"Action\": \"sqs:SendMessage\", \"Resource\": \"arn:aws:sqs:region:account-
id:queue-name\", \"Condition\": {\"ArnEquals\": {\"aws:SourceArn\":
 \"arn:aws:events:region:account-id:rule/rule-name\"}}}]}"
}
```

5. 使用剛建立的 set-queue-attributes.json 檔案作為輸入來設定政策屬性，如下列命令所示。

```
aws sqs set-queue-attributes \
```

```
--queue-url https://sqs.region.amazonaws.com/account-id/queue-name \
--attributes file://set-queue-attributes.json
```

如需詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的 Amazon SQS Policy Examples。

EventBridge 管道詳細資訊

EventBridge 管道不支援以資源為基礎的政策，且無支援以資源為基礎的政策條件的 API。

預防跨服務混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 AWS 中，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況，AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議在資源政策中使用 `aws:SourceArn` 或 `aws:SourceAccount` 全域條件內容索引鍵，來限制 Amazon EventBridge 給予另一項服務對資源的許可。如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 `aws:SourceArn`。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 `aws:SourceAccount`。

防範混淆代理人問題的最有效方法是使用 `aws:SourceArn` 全域條件內容索引鍵，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 `aws:SourceArn` 全域內容條件索引鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如：`arn:aws:servicename:*:123456789012:*`。

如果 `aws:SourceArn` 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須使用這兩個全域條件內容金鑰來限制許可。

事件匯流排

對於 EventBridge 事件匯流排規則目標，`aws:SourceArn` 的值必須是規則 ARN。

下列範例示範如何使用 EventBridge 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵，來預防混淆代理人問題。此範例適用於 EventBridge 規則所使用之角色的角色信任原則。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "events.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:events:*:123456789012:rule/myRule"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

EventBridge Pipes

對於 EventBridge Pipes，aws:SourceArn 的值必須是管道 ARN。

下列範例示範如何使用 EventBridge 中的 aws:SourceArn 和 aws:SourceAccount 全域條件內容索引鍵，來預防混淆代理人問題。此範例適用於角色信任原則，適用於 EventBridge 管道所使用的角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "events.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:pipe:*:123456789012::pipe/example"
      },
      "StringEquals": {
```

```
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Amazon EventBridge 結構的資源型政策

EventBridge 結構登錄檔支援資源型政策。資源型政策是連接到資源而非 IAM 身份的政策。例如，在 Amazon Simple Storage Service (Amazon S3) 中，資源政策會連接到 Amazon S3 儲存貯體。

如需 EventBridge 結構和資源型政策的詳細資訊，請參閱下列內容。

- Amazon EventBridge 結構 REST API 參考
- 《IAM 使用者指南》中的身份型政策和資源型政策

適用於資源型政策的支援的 API

您可以將下列 API 與資源型政策搭配用於 EventBridge 結構登錄檔。

- `DescribeRegistry`
- `UpdateRegistry`
- `DeleteRegistry`
- `ListSchemas`
- `SearchSchemas`
- `DescribeSchema`
- `CreateSchema`
- `DeleteSchema`
- `UpdateSchema`
- `ListSchemaVersions`
- `DeleteSchemaVersion`
- `DescribeCodeBinding`
- `GetCodeBindingSource`
- `PutCodeBinding`

將所有支援的動作授予 AWS 帳號的範例政策

對於 EventBridge 結構登錄檔，您必須始終將資源型政策附加到註冊表。若要授予對結構的存取權，請在政策中指定結構 ARN 和登錄 ARN。

若要授予使用者存取 EventBridge 結構的所有可用 API，請使用類似下列內容的政策，將 "Principal" 取代為您要授予存取權之帳戶的帳戶 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Test",
            "Effect": "Allow",
            "Action": [
                "schemas:*"
            ],
            "Principal": {
                "AWS": [
                    "109876543210"
                ]
            },
            "Resource": [
                "arn:aws:schemas:us-east-1:012345678901:registry/default",
                "arn:aws:schemas:us-east-1:012345678901:schema/default*"
            ]
        }
    ]
}
```

將唯讀動作授予 AWS 帳號的範例策略

下列範例僅會針對 EventBridge 結構的唯讀 API 授予帳戶的存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Test",
            "Effect": "Allow",
            "Action": [
                "schemas:DescribeRegistry",
                "schemas:ListSchemas",
                "schemas:SearchSchemas",
                "schemas:DescribeSchema",
                "schemas:ListSchemaVersions",
                "schemas:DescribeCodeBinding",
                "schemas:GetCodeBindingSource"
```

```
            ],
            "Principal": {
                "AWS": [
                    "109876543210"
                ]
            },
            "Resource": [
                "arn:aws:schemas:us-east-1:012345678901:registry/default",
                "arn:aws:schemas:us-east-1:012345678901:schema/default*"
            ]
        }
    ]
}
```

將所有動作授予組織的範例政策

您可以將資源型政策與 EventBridge 結構登錄檔搭配使用，以授予組織的存取權。如需詳細資訊，請參閱《AWS Organizations 使用者指南》。下列範例會授予具有結構登錄檔之 o-a1b2c3d4e5 存取權 ID 的組織。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Test",
            "Effect": "Allow",
            "Action": [
                "schemas:*"
            ],
            "Principal": "*",
            "Resource": [
                "arn:aws:schemas:us-east-1:012345678901:registry/default",
                "arn:aws:schemas:us-east-1:012345678901:schema/default*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": [
                        "o-a1b2c3d4e5"
                    ]
                }
            }
        }
    ]
```

```
}
```

Amazon EventBridge 許可參考

若要在 EventBridge 政策中指定動作，請使用 `events:` 字首後接 API 作業名稱，如下列範例所示。

```
"Action": "events:PutRule"
```

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": ["events:action1", "events:action2"]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，您可以指定名稱開頭有 `"Put"` 文字的所有動作，如下所示。

```
"Action": "events:Put*"
```

若要指定所有的 EventBridge API 動作，請使用 * 萬用字元，如下所示：

```
"Action": "events:*"
```

下表列出您可在 IAM 政策中指定的 EventBridge API 作業和對應動作。

EventBridge API 操作	所需的許可	描述
DeleteRule	`events:DeleteRule`	刪除規則時需要。
DescribeEventBus	`events:DescribeEventBus`	列出帳戶的必需項目，其可以將事件寫入目前帳戶的事件匯流排。
DescribeRule	`events:DescribeRule`	列出規則的詳細資訊時需要。
DisableRule	`events:DisableRule`	停用規則時需要。
EnableRule	`events:EnableRule`	啟用規則時需要。
ListRuleNamesByTarget	`events:ListRuleNamesByTarget`	列出與規則相關聯的目標時需要。

EventBridge API 操作	所需的許可	描述
ListRules	events:ListRules	列出您帳戶中的所有規則時需要。
ListTagsForResource	events:ListTagsForResource	列出與 EventBridge 資源相關聯的所有標籤時必填。目前，只能標記規則。
ListTargetsByRule	events:ListTargetsByRule	列出與規則相關聯的所有目標時需要。
PutEvents	events:PutEvents	新增可符合規則的自訂事件時需要。
PutPermission	events:PutPermission	必須提供另一個帳戶許可，以將事件寫入到此帳戶的預設事件匯流排。
PutRule	events:PutRule	建立或更新規則時需要。
PutTargets	events:PutTargets	將目標新增至規則時需要。
RemovePermission	events:RemovePermission	必須撤銷將事件寫入到此帳戶的預設事件匯流排的另一個帳戶許可權。
RemoveTargets	events:RemoveTargets	從規則中移除目標時需要。
TestEventPattern	events:TestEventPattern	針對特定事件測試事件模式時需要。

使用 IAM 政策條件進行精細定義存取控制

若要授與許可，請使用 IAM 政策語言指定政策生效時間的條件。例如，您可以在特定日期之後套用政策。

政策中的條件由索引鍵/值對所組成。條件索引鍵名稱不區分大小寫。

若您在單一條件中指定多個索引鍵，EventBridge 必須符合所有條件和索引鍵，EventBridge 才能授與權限。若您針對單一索引鍵使用多個值指定單一條件，EventBridge 會在符合其中一個值時授與權限。

您可以在指定條件時使用預留位置或政策變數。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/policyvariables.html 中的政策變數。如需使用 IAM 政策語言指定條件的詳細資訊，請參閱IAM 使用者指南中的條件。

在預設情況下，IAM; 使用者和角色無法存取您帳戶中的事件。若要使用事件，使用者必須獲得 `PutRule` API 動作的授權。如果您允許 IAM 使用者或角色執行其政策的 `events:PutRule` 動作，則他們可以建立符合特定事件的規則。但是，為了使規則有用，使用者還必須具有 `events:PutTargets` 動作的權限，因為如果您希望規則執行的不僅僅是發佈 CloudWatch 指標，則還必須將目標新增至規則。

您可以在 IAM 使用者或角色的政策陳述式中提供條件，讓他們建立僅符合一組特定來源和事件類型的規則。若要授與特定來源和事件類型的存取權，請使用 `events:source` 和 `events:detail-type` 條件索引鍵。

類似地，您可以在 IAM 使用者或角色的政策陳述式中提供條件，讓他們在您的帳戶中建立來源。要授予對特定資源的訪問權限，請使用 `events:TargetArn` 條件鍵。

下列範例是一項政策，可讓使用者使用 `PutRule` API 動作的拒絕陳述式存取 EventBridge 中的 Amazon EC2 事件以外的所有事件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPutRuleForAllEC2Events",
            "Effect": "Deny",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:source": "aws.ec2"
                }
```

```
            }
        }
    ]
}
```

EventBridge 條件鍵

下表顯示您可以在 EventBridge 政策中使用的條件索引鍵以及鍵和值配對。

條件鍵	鍵值對	評估類型
aws:SourceAccount	由 `aws:SourceArn` 指定的規則存在的帳戶。	Account Id, Null
aws:SourceArn	傳送事件的規則的 ARN。	ARN, Null
events:creatorAccount	`"events:creatorAccount":" creatorAccount "` 若為`####`，請針對建立規則的帳戶使用帳戶 ID。使用此條件可針對特定帳戶的規則授權 API 呼叫。	creatorAccount, Null
events:detail-type (事件:詳細資料類型)	`"events:detail-type" detail-type "` 其中的 *`detail-type`* 是事件的 detail-type 欄位的文字字串，例如 `"AWS API Call via CloudTrail"` 和 `"EC2 Instance State-change Notification"` 。	Detail Type, Null
events: detail.eventTypeCode	`"events:detail.eventTypeCode":" eventTypeCode "` 對於 *`eventTypeCode`* ，使用 detail.eventTypeCode 欄位	eventTypeCode，Null

條件鍵	鍵值對	評估類型
	的文字字串，例如 "AWS_ABUSE_DOS_REPORT" 。	
events: detail.service	"events:detail.service":" *service*" 對於*##*，使用事件的 detail.service 欄位的文字字串，例如 "ABUSE"。	服務，Null
events: detail.userIdentity.principalId	"events:detail.userIdentity.principalId":" *principal-id* " 對於 *principal-id* 使用 detail-type "AWS API Call via CloudTrail" 事件的 detail.userIdentity.principalId 欄位的文字字串，例如"AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName." 。	Principal Id，Null
events：eventBusInvocation	"events:eventBusInvocation":" *boolean*" 對於*###*，當規則將事件傳送至另一個帳戶中事件匯流排的目標時，請使用 true。使用 PutEvents API 呼叫時使用 false。	eventBusInvocation, Null
events:ManagedBy	由 AWS 服務在內部使用。如果規則是由代表你的 AWS 服務所建立，則該值為建立規則之服務主體之名稱。	不適用於客戶政策。

條件鍵	鍵值對	評估類型
events:source	"events:source":" *source* " 使用##作為事件的來源欄位的文字字串，例如 "aws.ec2" 和 "aws.s3"。若要查看更多可能的##值，請參閱 來自 AWS 服務的事件 中的範例事件。	Source，Null
events:TargetArn	"events:TargetArn" :" *target-arn* " 對於*Target arn*，請使用規則的目標的 ARN，例如 "arn:aws:lambda:*:*:function:*" 。	ArrayOfARN, Null

如需 EventBridge 的範例政策陳述式，請參閱 管理對您 Amazon EventBridge 資源的存取許可。

主題

- EventBridge 管道細節
- 範例：使用 creatorAccount 條件
- 範例：使用 eventBusInvocation 條件
- 範例：限制存取特定資源
- 範例：定義多個可個別用於事件模式的來源
- 範例：定義可用於事件模式的來源和 DetailType
- 範例：確定事件模式中的來源已定義
- 範例：在具有多個來源的事件模式中定義允許來源清單
- 範例：限制 PutRule 存取 detail.service
- 範例：限制 PutRule 存取 detail.eventTypeCode
- 範例：確保僅允許來自某個特定 PrincipalId API 呼叫的 AWS CloudTrail 事件
- 範例：限制存取目標

EventBridge 管道細節

EventBridge 管道不支援任何其他 IAM 政策條件鍵。

範例：使用 **creatorAccount** 條件

下列範例策略陳述式顯示如何使用策略中的 creatorAccount 條件，以便僅在指定為的帳號 creatorAccount 是建立規則的帳號時，才允許建立規則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleForOwnedRules",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "events:creatorAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
}
```

範例：使用 **eventBusInvocation** 條件

eventBusInvocation 指示調用是否來自跨帳戶目標還是 PutEvents API 要求。如果調用來自包含跨帳戶目標的規則 (例如目標是另一個帳戶中的事件匯流排)，則此值為 true。當調用來自 PutEvents API 請求的結果時，該值為 false。下列範例指出跨帳戶目標的調用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossAccountInvocationEventsOnly",
      "Effect": "Allow",
      "Action": "events:PutEvents",
      "Resource": "*",
      "Condition": {
```

```
        "BoolIfExists": {
          "events:eventBusInvocation": "true"
        }
      }
    }
  ]
}
```

範例：限制存取特定資源

以下範例政策可連接至 IAM; 使用者。政策 A 允許所有事件的 PutRule API 動作，而政策 B 只有在所建立規則的事件模式符合 EC2 事件時，才允許 PutRule。

政策 A：允許所有事件

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleForAllEvents",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*"
        }
    ]
    }
```

政策 B：允許來自 EC2 的事件

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleForAllEC2Events",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:source": "aws.ec2"
                }
            }
```

```
        }
    ]
}
```

EventPattern 是 PutRule 的必要引數。因此，如果具有政策 B 的使用者以類似以下事件的模式呼叫 PutRule，

```
{
    "source": [ "aws.ec2" ]
}
```

則系統會建立規則，因為政策允許此特定來源，亦即 "aws.ec2"。不過，如果具有政策 B 的使用者以類似以下事件的模式呼叫 PutRule，則會拒絕建立規則，因為政策不允許此特定來源：也就是 "aws.s3"。

```
{
    "source": [ "aws.s3" ]
}
```

實際上，擁有政策 B 的使用者僅允許建立符合源自於 Amazon EC2 事件的規則；因此，僅允許這些使用者存取來自 Amazon EC2 的事件。

請參閱下表以取得政策 A 和政策 B 的比較。

事件模式	政策 A 允許	政策 B 允許
`{ "source": [ "aws.ec2" ] }`	是	是
`{ "source": [ "aws.ec2", "aws.s3" ] }`	是	否 (不允許 aws.s3 Source)
{	是	是

事件模式	政策 A 允許	政策 B 允許

<pre>"source":
["aws.ec2"],
"detail-type":
["EC2 Instance
State-change
Notification"]
}</pre>		

<pre>{
"detail-type":
["EC2 Instance
State-change
Notification"]
}</pre>
	是	否 (必須指定來源)

範例：定義多個可個別用於事件模式的來源

以下政策允許 IAM 使用者或角色建立規則，其中的 `EventPattern` 來源為 Amazon EC2 或 Amazon ECS。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleIfSourceIsEC2OrECS",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:source": [ "aws.ec2", "aws.ecs" ]
                }
            }
        }
    ]
}
```

下表展示了此政策允許或拒絕的事件模式範例。

事件模式	政策允許
`{ "source": [ "aws.ec2" ] }`	是
`{ "source": [ "aws.ecs" ] }`	是
`{ "source": [ "aws.s3" ] }`	否
`{ "source": [ "aws.ec2", "aws.ecs" ] }`	否
`{ "detail-type": [ "AWS API Call via CloudTrail" ] }`	否

範例：定義可用於事件模式的來源和 DetailType

以下政策僅允許來自 aws.ec2 來源且 DetailType 等於 EC2 instance state change notification 的事件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid":
 "AllowPutRuleIfSourceIsEC2AndDetailTypeIsInstanceStateChangeNotification",
            "Effect": "Allow",
            "Action": "events:PutRule",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:source": "aws.ec2",
                    "events:detail-type": "EC2 Instance State-change Notification"
                }
            }
        }
    ]
}
```

下表展示了此政策允許或拒絕的事件模式範例。

事件模式	政策允許
`{ "source": [ "aws.ec2" ] }`	否
`{ "source": [ "aws.ecs" ] }`	否
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance State-change Notification" ] }`	是
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance Health Failed" ] }`	否
`{`	否

事件模式	政策允許

<pre>"detail-type": ["EC2 Instance State-change Notification"]
}</pre>	

範例：確定事件模式中的來源已定義

以下政策允許用戶僅以 EventPatterns (必須具有來源欄位) 建立規則。換言之，IAM 使用者或角色無法以未提供特定來源的 EventPattern 建立規則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleIfSourceIsSpecified",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "events:source": "false"
                }
            }
        }
    ]
}
```

下表展示了此政策允許或拒絕的事件模式範例。

事件模式	政策允許

<pre>{
"source": ["aws.ec2"],
"detail-type": ["EC2 Instance State-change Notification"]
}</pre>
	是

事件模式	政策允許

<pre>{
"source": ["aws.ecs",
"aws.ec2"]
}</pre>
	是

<pre>{
"detail-type": ["EC2
Instance State-change Notificat
ion"]
}</pre>
	否

範例：在具有多個來源的事件模式中定義允許來源清單

以下政策允許以 EventPatterns (可擁有多個來源) 建立規則。事件模式中列出的每個來源都必須是條件中提供的清單成員。當使用 ForAllValues 條件時，請確定條件清單中至少有一個項目已經定義。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleIfSourceIsSpecifiedAndIsEitherS3OrEC2OrBoth",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "events:source": [ "aws.ec2", "aws.s3" ]
                },
                "Null": {
                    "events:source": "false"
                }
            }
        }
    ]
}
```

下表展示了此政策允許或拒絕的事件模式範例。

事件模式	政策允許
`{ "source": [ "aws.ec2" ] }`	是
`{ "source": [ "aws.ec2", "aws.s3" ] }`	是
`{ "source": [ "aws.ec2", "aws.autoscaling" ] }`	否
`{ "detail-type": [ "EC2 Instance State-change Notification" ] }`	否

範例：限制 **PutRule** 存取 **detail.service**

您可以將 IAM 使用者或角色限制為僅為 `events:details.service` 欄位中具有特定值的事件建立規則。`events:details.service` 的值並不一定是 AWS 服務的名稱。

此政策條件可協助處理與安全性或濫用相關的 AWS Health 事件。透過此政策條件，您可以限制只有需要的使用者才能夠查看這些敏感提醒的存取權。

例如，以下政策允許僅為 `events:details.service` 值是 ABUSE 的事件建立規則。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowPutRuleEventsWithDetailServiceEC2",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:detail.service": "ABUSE"
                }
            }
        }
    ]
}
```

範例：限制 **PutRule** 存取 **detail.eventTypeCode**

您可以將 IAM 使用者或角色限制為僅為 events:details.eventTypeCode 欄位中具有特定值的事件建立規則。此政策條件可協助處理與安全性或濫用相關的 AWS Health 事件。透過此政策條件，您可以限制只有需要的使用者才能夠查看這些敏感提醒的存取權。

例如，以下政策允許僅為 events:details.eventTypeCode 值是 AWS_ABUSE_DOS_REPORT 的事件建立規則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleEventsWithDetailServiceEC2",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:detail.eventTypeCode": "AWS_ABUSE_DOS_REPORT"
                }
            }
        }
    ]
}
```

範例：確保僅允許來自某個特定 **PrincipalId** API 呼叫的 AWS CloudTrail 事件

所有 AWS CloudTrail 事件都有使用者 ID，該使用者在事件的 `detail.userIdentity.principalId` 路徑執行 API 呼叫 。利用 `events:detail.userIdentity.principalId` 條件鍵，您可以限制 IAM 使用者或角色僅能存取來自特定帳戶的 CloudTrail 事件。

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleOnlyForCloudTrailEventsWhereUserIsASpecificIAMUser",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:detail-type": [ "AWS API Call via CloudTrail" ],
                    "events:detail.userIdentity.principalId":
 [ "AIDAJ45Q7YFFAREXAMPLE" ]
                }
            }
        }
    ]
}
```

下表展示了此政策允許或拒絕的事件模式範例。

事件模式	政策允許
`{ "detail-type": [ "AWS API Call via CloudTrail" ] }`	否
`{ "detail-type": [ "AWS API Call via CloudTrail" ], "detail.userIdentity.principalId": [ "AIDAJ45Q7YFFAREXAMPLE" ]`	是

事件模式	政策允許
}	
{ "detail-type": ["AWS API Call via CloudTrail"], "detail.userIdentity.principalId": ["AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName"] }	否

範例：限制存取目標

如果 IAM; 使用者或角色具有 events:PutTargets 許可，他們可以將相同帳戶中的任何目標新增至他們允許存取的規則。以下政策限制只能將目標新增至特定規則 (帳戶 123456789012 下的 MyRule)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRule",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule"
        }
    ]
}
```

若要限制哪些目標可以新增至規則，請使用 events:TargetArn 條件鍵。您可以將目標限制為只有 Lambda 函數，如下列範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRuleAndOnlyLambdaFunctions",
            "Effect": "Allow",
```

```
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule",
            "Condition": {
                "ArnLike": {
                    "events:TargetArn": "arn:aws:lambda:*:*:function:*"
                }
            }
        }
    ]
}
```

使用 EventBridge 的服務連結角色

Amazon EventBridge 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 EventBridge 的一種特殊 IAM 角色類型。服務連結角色由 EventBridge 預先定義，且內含該服務代您呼叫其他 AWS 服務所需的所有許可。

主題

- 使用角色為 API 目的地建立密碼
- 使用角色來探索結構描述

使用角色為 API 目的地建立密碼

Amazon EventBridge 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 EventBridge 的一種特殊 IAM 角色類型。服務連結角色由 EventBridge 預先定義，且內含該服務代您呼叫其他 AWS 服務所需的所有許可。

服務連結的角色可讓設定 EventBridge 更為簡單，因為您不必手動新增必要的許可。EventBridge 定義其服務連結角色的許可，除非另有定義，否則僅有 EventBridge 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此可保護您 EventBridge 的資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的 AWS 服務》，尋找 Service-linked roles (服務連結角色) 欄中顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

EventBridge 的服務連結角色許可

EventBridge使用名為的服務連結角色 AWSServiceRoleForAmazonEventBridgeApiDestinations— 允許存取由EventBridge建立的 Secret 管理員密碼。

AWSServiceRoleForAmazonEventBridgeApiDestinations 服務連結角色信任下列服務以擔任角色：

- `apidestinations.events.amazonaws.com`

名為 Policy 的角色權限AmazonEventBridgeApiDestinationsServiceRole原則EventBridge允許對指定的資源完成下列動作：

- 動作：`secrets created for all connections by EventBridge` 上的 `create, describe, update and delete secrets; get and put secret values`

您必須設定許可，以允許您的使用者、群組或角色建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

為 EventBridge 建立服務連結角色

您不需要手動建立一個服務連結角色。當您在AWS Management Console、或 AWS API 中建EventBridge立連線時AWS CLI，會為您建立服務連結角色。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。如果您在 2021 年 2 月 11 日之前使用EventBridge服務，則該服務開始支援服務連結角色時，請在您的帳戶中EventBridge建立該AWSServiceRoleForAmazonEventBridgeApiDestinations角色。若要進一步了解，請參閱在我的 AWS 帳戶 中顯示新角色。

若您刪除此服務連結角色然後需要再次建立，便可在帳戶中使用相同程序重新建立角色。建立連線時，請再次為您建EventBridge立服務連結角色。

為 EventBridge 編輯服務連結角色

EventBridge 不允許您編輯 AWSServiceRoleForAmazonEventBridgeApiDestinations 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

為 EventBridge 刪除服務連結角色

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，務必清除您的服務連結角色，之後才能以手動方式將其刪除。

清除服務連結角色

在您使用 IAM 刪除服務連結角色之前，您必須先刪除該角色所使用的任何資源。

> **Note**
>
> 若 EventBridge 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除 AWSServiceRoleForAmazonEventBridgeApiDestinations 所使用的 EventBridge 資源 (主控台)

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在「整合」下選擇「API 目標」，然後選擇「連線」標籤。
3. 選擇連線，然後選擇「刪除」。

刪除 AWSServiceRoleForAmazonEventBridgeApiDestinations 所使用的 EventBridge 資源 (AWS CLI)

- 使用以下命令：`delete-connection`。

刪除 AWSServiceRoleForAmazonEventBridgeApiDestinations 所使用的 EventBridge 資源 (API)

- 使用以下命令：`DeleteConnection`。

手動刪除服務連結角色

使用 IAM 主控台、AWS CLI 或 AWS API 來刪除 AWSServiceRoleForAmazonEventBridgeApiDestinations 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

EventBridge 服務連結角色的支援區域

EventBridge 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AWS區域和端點。

使用角色來探索結構描述

Amazon EventBridge 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 EventBridge 的一種特殊 IAM 角色類型。服務連結角色由 EventBridge 預先定義，且內含該服務代您呼叫其他 AWS 服務所需的所有許可。

服務連結的角色可讓設定 EventBridge 更為簡單，因為您不必手動新增必要的許可。EventBridge 定義其服務連結角色的許可，除非另有定義，否則僅有 EventBridge 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此可保護您 EventBridge 的資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的 AWS 服務》，尋找 Service-linked roles (服務連結角色) 欄中顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

EventBridge 的服務連結角色許可

EventBridge使用名為 AWSServiceRoleForSchemas— 授與由結Amazon EventBridge構描述建立的受管規則的權限的服務連結角色。

AWSServiceRoleForSchemas 服務連結角色信任下列服務以擔任角色：

- `schemas.amazonaws.com`

名為的角色權限原則AmazonEventBridgeSchemasServiceRolePolicyEventBridge允許對指定的資源完成下列動作：

- 動作：`all managed rules created by EventBridge` 上的 `put, enable, disable, and delete rules; put and remove targets; list targets per rule`

您必須設定許可，以允許您的使用者、群組或角色建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

為 EventBridge 建立服務連結角色

您不需要手動建立一個服務連結角色。當您在AWS Management Console、或 AWS API 中執行結構描述探索時AWS CLI，會為您EventBridge建立服務連結角色。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。如果您在 2019 年 11 月 27 日之前使用該EventBridge服務，則該服務開始支援服務連結角色時，請在您的帳戶中EventBridge建立該AWSServiceRoleForSchemas角色。若要進一步了解，請參閱在我的 AWS 帳戶 中顯示新角色。

若您刪除此服務連結角色然後需要再次建立，便可在帳戶中使用相同程序重新建立角色。當您執行結構描述探索時，請再次為您EventBridge建立服務連結角色。

為 EventBridge 編輯服務連結角色

EventBridge 不允許您編輯 AWSServiceRoleForSchemas 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

為 EventBridge 刪除服務連結角色

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，務必清除您的服務連結角色，之後才能以手動方式將其刪除。

清除服務連結角色

在您使用 IAM 刪除服務連結角色之前，您必須先刪除該角色所使用的任何資源。

> **Note**
>
> 若 EventBridge 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除 AWSServiceRoleForSchemas 所使用的 EventBridge 資源 (主控台)

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 在「巴士」下，選擇「活動巴士」，然後選擇活動巴士。
3. 選擇 [停止探索]。

刪除 AWSServiceRoleForSchemas 所使用的 EventBridge 資源 (AWS CLI)

- 使用以下命令：`delete-discoverer`。

刪除 AWSServiceRoleForSchemas 所使用的 EventBridge 資源 (API)

- 使用以下命令：`DeleteDiscoverer`。

手動刪除服務連結角色

使用 IAM 主控台、AWS CLI 或 AWS API 來刪除 AWSServiceRoleForSchemas 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

EventBridge 服務連結角色的支援區域

EventBridge 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AWS區域和端點。

使用記錄 Amazon EventBridge API 呼叫 AWS CloudTrail

Amazon EventBridge 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 擷取 EventBridge 作為事件的所有 API 呼叫。擷取的呼叫包括來自 EventBridge 主控台的呼叫和 EventBridge API 作業的程式碼呼叫。使用收集的資訊 CloudTrail，您可以判斷提出的要求 EventBridge、提出要求的 IP 位址、提出要求的時間，以及其他詳細資訊。

每一筆事件或日誌項目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。

EventBridge 資料事件 CloudTrail

資料事件提供在資源上或在資源中執行的資源操作的相關資訊 (例如，讀取或寫入 Amazon S3 物件)。這些也稱為資料平面操作。資料事件通常是大量資料的活動。依預設， CloudTrail 不會記錄資料事件。 CloudTrail 事件歷史記錄不會記錄數據事件。

資料事件需支付額外的費用。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。

您可以使用 CloudTrail 主控台或 CloudTrail API 作業記錄 EventBridge 資源類型的資料事件。 AWS CLI有關如何記錄資料事件的詳細資訊，請參閱AWS CloudTrail 使用《使用指南》 AWS Command Line Interface中的記錄資料事件 AWS Management Console和記錄資料事件。

下表列出您可以記錄 EventBridge 資料事件的資源類型。[資料事件類型 (主控台)] 欄顯示可從主控台的 [資料事件類型 CloudTrail] 清單中選擇的值。resource .type 值欄會顯示**`resources.type`**值，您可以在使用或 API 設定進階事件選取器時指定這個值。 AWS CLI CloudTrail 記錄到資料 CloudTrail欄中的資料 API 會顯示 CloudTrail 針對資源類型記錄的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
活動巴士	`AWS::Events::EventBus`	• DescribeEventBus
活動總線規則	`AWS::Events::Rule`	• DeleteRule • DescribeRule • DisableRule • EnableRule • ListRuleNamesByTarget • ListRules • ListTargetsByRule • PutRule

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
		• PutTargets • RemoveTargets • TestEventPattern
管	AWS::Pipes::Pipe	• CreatePipe • DeletePipe • DescribePipe • ListPipes • StartPipe • StopPipe • UpdatePipe

您可以設定進階事件選取器來篩選eventNamereadOnly、和resources.ARN欄位，以僅記錄對您很重要的事件。如需這些欄位的詳細資訊，請參閱 AWS CloudTrail API 參考AdvancedFieldSelector中的。

EventBridge 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。

Amazon EventBridge 將所有 EventBridge 控制平面作業記錄為管理事件。如需記 EventBridge 錄到的 Amazon EventBridge 控制平面作業清單 CloudTrail，請參閱 Amazon EventBridge API 參考。

EventBridge 事件範例

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PutRule作業的 CloudTrail 事件。

```
{
  "eventVersion":"1.03",
  "userIdentity":{
    "type":"Root",
    "principalId":"123456789012",
```

```
    "arn":"arn:aws:iam::123456789012:root",
    "accountId":"123456789012",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "sessionContext":{
      "attributes":{
      "mfaAuthenticated":"false",
      "creationDate":"2015-11-17T23:56:15Z"
      }
    }
  },
  "eventTime":"2015-11-18T00:11:28Z",
  "eventSource":"events.amazonaws.com",
  "eventName":"PutRule",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"AWS Internal",
  "userAgent":"AWS CloudWatch Console",
  "requestParameters":{
    "description":"",
    "name":"cttest2",
    "state":"ENABLED",
    "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-
change Notification\"]}",
    "scheduleExpression":""
  },
  "responseElements":{
    "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
  },
  "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
  "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
  "eventType":"AwsApiCall",
  "apiVersion":"2015-10-07",
  "recipientAccountId":"123456789012"
}
```


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。

CloudTrail EventBridge 管道採取的動作的記錄項目

EventBridge 從來源讀取事件、叫用擴充或叫用目標時，管道會採用提供的 IAM 角色。對於在您的帳戶中對所有擴充、目標以及 Amazon SQS、Kinesis 和 DynamoDB 來源採取的動作相關的 CloudTrail 項目，和欄位將包括在`sourceIPAddress`內。`invokedBy` `pipes.amazonaws.com`

適用於所有擴充、目標以及 Amazon SQS、Kinesis 動和 DynamoDB 來源的範例 CloudTrail 日誌項目

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "...",
    "arn": "arn:aws:sts::111222333444:assumed-role/...",
    "accountId": "111222333444",
    "accessKeyId": "...",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "...",
        "arn": "...",
        "accountId": "111222333444",
        "userName": "userName"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2022-09-22T21:41:15Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "pipes.amazonaws.com"
  },
  "eventTime": ",,,",
  "eventName": "...",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "pipes.amazonaws.com",
  "userAgent": "pipes.amazonaws.com",
  "requestParameters": {
    ...
  },
  "responseElements": null,
  "requestID": "...",
  "eventID": "...",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "...",
  "eventCategory": "Management"
}
```

對於所有其他來源， CloudTrail 記錄項目的`sourceIPAddress`欄位將具有動態 IP 位址，而且不應將其用於任何整合或事件分類。此外，這些條目將沒有該 `invokedBy` 字段。

所有其他來源的範例 CloudTrail 記錄項目

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    ...
  },
  "eventTime": ",,,",
  "eventName": "...",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "Python-httplib2/0.8 (gzip)",
}
```

Amazon EventBridge 的合規驗證

在多個 AWS 合規計畫中，第三方稽核人員 (例如：SOC、PCI、FedRAMP 和 HIPAA) 會評估 AWS 服務的安全性與合規性。

如需特定合規計畫的 AWS 服務範圍清單，請參閱合規計畫的 AWS 服務範圍。如需一般資訊，請參閱 AWS 合規計劃 。

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

您使用 EventBridge 時的合規責任取決於資料的敏感度、您公司的合規目標，以及適用的法律和法規。AWS 提供以下資源協助您處理合規事宜：

- 安全性與合規快速入門指南：這些部署指南會描述架構考量，並提供在 AWS 上部署以安全性及合規為重心之基準環境的步驟。
- HIPAA 安全與合規架構白皮書：公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源 ：一組手冊和指南。
- 《AWS Config 開發人員指南》中的使用規則評估資源：有關 AWS Config 如何評估資源組態對於內部實務、業界準則和法規的合規狀態的資訊。
- AWS Security Hub：全面檢視您 AWS 中的安全狀態，可助您檢查是否符合安全產業標準和最佳實務。

Amazon EventBridge 彈性

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，並以低延遲、高輸送量和高度備援聯網功能相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，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。

如需 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

Amazon EventBridge 中的基礎設施安全

Amazon EventBridge 是一項受管服務，受 AWS 全球網路安全保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可使用 AWS 發佈的 API 呼叫來透過網路存取 EventBridge。用戶端必須支援下列項目：

- 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，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可從任何網路位置呼叫這些 API 操作，且能夠在 EventBridge 使用資源型存取政策，這類政策納入的限制可針對來源 IP 地址。您也可以使用 EventBridge 政策來控制 Amazon Virtual Private Cloud (Amazon VPC) 端點或特定 VPC 的存取權。實際上，這就將特定 EventBridge 資源的網路存取與 AWS 網路內的特定 VPC 隔離開來。

Amazon EventBridge 中的組態與漏洞分析

組態和 IT 控制是 AWS 與身為我們客戶的您共同的責任。如需詳細資訊，請參閱 AWS 共同的責任模型。

監控 Amazon EventBridge

EventBridge 從符合事件的數量到規則叫用目標的次數， CloudWatch 每分鐘都會傳送指標至 Amazon。

下列影片可透過以下方式檢閱監控和稽核 EventBridge 行為 CloudWatch：監控和稽核事件

主題

- EventBridge 度量
- 量度的維 EventBridge 度

EventBridge 度量

AWS/Events 命名空間包含下列指標。

對於使用 Count 作為單位的指標，Sum 並且 SampleCount 往往是最有用的統計信息。

僅指定RuleName維度的量度會參照預設事件匯流排。同時指定EventBusName和RuleName維度的量度會參照自訂事件匯流排。

指標	描述
DeadLetterInvocations	規則目標未被調用以對事件進行回應時的次數。這包含會再次運行相同規則、導致無限迴圈的調用。 有效尺寸: RuleName 單位：計數
Events	由所 EventBridge擷取的合作夥伴事件數目。 有效尺寸: EventSourceName 單位：計數
FailedInvocations	永久失敗的調用次數。這不包括重試或嘗試重試後成功的調用。它亦不會將計入 DeadLetterInvocations 的失敗調用列入計算。

指標	描述
	Note EventBridge 只有在不為零時 CloudWatch ，才會將此量度傳送至。 有效尺寸: RuleName 單位：計數
Invocations	規則回應事件時調用目標的次數。這包含成功與失敗的調用，但不包含永久失敗前的節流或重試嘗試。它不包括 DeadLetterInvocations 。 Note EventBridge 只有在不為零時 CloudWatch ，才會將此量度傳送至。 有效尺寸：無 RuleName 單位：計數
InvocationAttempts	EventBridge 嘗試呼叫目標的次數。 有效維度：無 單位：計數
InvocationsCreated	為回應每個事件而建立的調用總數。 此測量結果通常用來監督每秒EventBridge 服務配額交易中「呼叫」節流限制的使用率。 有效維度：無 單位：計數

指標	描述
InvocationsFailedToBeSentToDlq	無法移至無效字母佇列的調用數量。無效字母佇列錯誤可能因許可錯誤、設定錯誤的資源或大小限制而發生。 Note EventBridge 只有在不為零時 CloudWatch ，才會將此量度傳送至。 有效尺寸: RuleName 單位：計數
IngestionToInvocationCompleteLatency	從事件擷取到第一次成功調用嘗試完成所花費的時間。 有效尺寸： EventBusName，無， RuleName 單位：毫秒
IngestionToInvocationStartLatency	處理事件的時間，從擷取事件 EventBridge 到目標的第一次叫用開始測量。 有效尺寸： EventBusName，無， RuleName 單位：毫秒
InvocationsSentToDlq	移至無效字母佇列的調用數目。 Note EventBridge 只有在不為零時 CloudWatch ，才會將此量度傳送至。 有效尺寸: RuleName 單位：計數

指標	描述
MatchedEvents	如果指 EventSourceName 定 EventBusName 或，則表示與任何規則相符的事件數目。如果 RuleName 指定，則表示與特定規則相符的事件數目。 有效尺寸: EventBusName, RuleName, EventSourceName 單位：計數
RetryInvocationAttempts	重試目標調用的次數。 Note EventBridge 只有在不為零時 CloudWatch ，才會將此量度傳送至。 有效維度：無 單位：計數
SuccessfulInvocationAttempts	成功調用目標的次數。 有效維度：無 單位：計數
ThrottledRules	規則執行限流的次數。這些規則的調用可能會延遲。 如需詳細資訊，請參閱 ??? 中每秒交易的調用限流。 有效尺寸： EventBusName，無， RuleName 單位：計數

指標	描述
TriggeredRules	已執行並符合任何事件的規則數目。 在觸發規則之前，您不會在中看 CloudWatch 到此量度。 有效尺寸： EventBusName，無， RuleName 單位：計數

EventBridge PutEvents 度量

AWS/Events 命名空間包含有關 PutEvents API 請求的下列要求指標。

對於使用 Count 作為單位的指標，Sum 並且 SampleCount 往往是最有用的統計信息。

指標	描述
PutEvents ApproximateCallCount	收到的 PutEvents 請求的大約數目。 有效維度：無 單位：計數
PutEvents ApproximateFailedCount	失敗的 PutEvents 請求的大約數目。 有效維度：無 單位：計數
PutEvents ApproximateSuccessCount	成功的 PutEvents 請求的大約數目。 有效維度：無 單位：計數
PutEvents ApproximateThrottledCount	因限流而拒絕的 PutEvents 請求的數目。 有效維度：無

指標	描述
	單位：計數
PutEvents EntriesCount	PutEvents 請求中包含的事件項目數目。 有效維度：無 單位：計數
PutEvents FailedEntriesCount	無法擷取之 PutEvents 請求中包含的事件項目數目。 有效維度：無 單位：計數
PutEvents Latency	每個 PutEvents 請求所花費的時間。 有效維度：無 單位：毫秒
PutEvents RequestSize	PutEvents 請求的大小。 有效維度：無 單位：位元組

EventBridge PutPartnerEvents 度量

AWS/Events 命名空間包含有關 PutPartnerEvents API 請求的下列要求指標。

> **Note**
>
> EventBridge 僅包含與傳送事件的 SaaS 合作夥伴帳戶中PutPartnerEvents要求相關的量度。如需更多資訊，請參閱???

對於使用 Count 作為單位的指標，Sum 並且 SampleCount 往往是最有用的統計信息。

指標	描述
PutPartnerEventsApproximateCallCount	收到的 PutPartnerEvents 請求的大約數目。 有效維度：無 單位：計數
PutPartnerEventsApproximateFailedCount	失敗的 PutPartnerEvents 請求的大約數目。 有效維度：無 單位：計數
PutPartnerEventsApproximateThrottledCount	因限流而拒絕的 PutPartnerEvents 請求的數目。 有效維度：無 單位：計數
PutPartnerEventsApproximateSuccessCount	成功的 PutPartnerEvents 請求的大約數目。 有效維度：無 單位：計數
PutPartnerEventsEntriesCount	PutPartnerEvents 請求中包含的事件項目數目。 有效維度：無 單位：計數
PutPartnerEventsFailedEntriesCount	無法擷取之 PutPartnerEvents 請求中包含的事件項目數目。 有效維度：無 單位：計數
PutPartnerEventsLatency	每個 PutPartnerEvents 請求所花費的時間。 有效維度：無

指標	描述
	單位：毫秒

量度的維 EventBridge 度

EventBridge 量度具有維度或可排序屬性，如下所示。

維度	描述
EventBusName	依據事件匯流排名稱篩選可用的指標。
EventSourceName	依據合作夥伴事件來源名稱篩選可用的指標。
RuleName	依據規則名稱篩選可用的指標。

Amazon 故障 EventBridge

您可以使用本節中的步驟對 Amazon 進行疑難排解 EventBridge。

主題

我的規則可以運行，但是我的 Lambda 函數未被調用

您的 Lambda 函數可能無法執行的原因之一是您未擁有正確的許可。

檢查 Lambda 函數的許可

1. 使用 AWS CLI，使用函數和您的 AWS 區域執行下列命令：

```
aws lambda get-policy --function-name MyFunction --region us-east-1
```

您應該會看到下列輸出。

```
{
    "Policy": "{\"Version\":\"2012-10-17\",
    \"Statement\":[
        {\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:events:us-
east-1:123456789012:rule/MyRule\"}},
        \"Action\":\"lambda:InvokeFunction\",
        \"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:MyFunction\",
        \"Effect\":\"Allow\",
        \"Principal\":{\"Service\":\"events.amazonaws.com\"},
        \"Sid\":\"MyId\"}
    ],
    \"Id\":\"default\"}"
}
```

2. 如果您看到以下錯誤訊息。

```
A client error (ResourceNotFoundException) occurred when calling the GetPolicy
 operation: The resource you requested does not exist.
```

或者，您有看到輸出，但無法將 events.amazonaws.com 配置為政策中的信任實體，請執行下列命令：

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/MyRule
```

3. 如果輸出包含 SourceAccount 欄位，那麼您需要將其移除。SourceAccount設定可防 EventBridge 止呼叫函數。

> **Note**
>
> 如果原則不正確，您可以在 EventBridge 主控台中編輯規則，方法是移除規則，然後再將其新增回規則。接著， EventBridge 主控台會在目標上設定正確的權限。
> 如果您使用特定的 Lambda 別名或版本，如下列命令所示，您必須將 `--qualifier` 參數新增至 `aws lambda get-policy` 和 `aws lambda add-permission` 命令。

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/MyRule
--qualifier alias or version
```

我剛建立或修改規則，但不符合測試事件

當您變更規則或其目標，連入事件可能不會立即開始或停止比對新的或更新的規則。允許一小段時間來讓變更生效。

如果短時間後事件仍然不匹配，請檢查指 CloudWatch 標 `TriggeredRulesInvocations`，以及您`FailedInvocations`的規則。如需這些指標的詳細資訊，請參閱監控 Amazon EventBridge。

如果規則的目的是要比對來自 AWS 服務的事件，請執行下列其中一項作業：

- 使用此 `TestEventPattern` 動作可測試規則與測試事件相符的事件模式。如需詳細資訊，請TestEventPattern參閱 Amazon EventBridge API 參考中的。
- 使用主EventBridge 控台上的「沙箱」。

我的規則在 `ScheduleExpression` 中我指定的時間沒有運行

請確定您已設定規則在 UTC+0 時區的排程。如果 `ScheduleExpression` 正確，請按照 我剛建立或修改規則，但不符合測試事件 中的步驟動作。

我的規則並未在我預期的時間運行

EventBridge 在您設定的開始時間的一分鐘內執行規則。一旦您建立規則，執行時間的倒數計數將立即開始。

> **Note**
>
> 已排程規則具有 `guaranteed` 意義事件的交付類型，每個預期的時間至少會觸發一次。

您可以使用 cron 表達式在指定的時間調用目標。若要建立在第 0 分鐘每四小時執行一次的規則，請執行下列其中一項：

- 在 EventBridge 主控台中，您可以使用 cron 運算式`0 0/4 * * ? *`。
- 使用 AWS CLI，您可以使用運算式`cron(0 0/4 * * ? *)`。

例如，若要建立名`TestRule`為每 4 小時執行一次的規則 AWS CLI，請使用下列命令。

```
aws events put-rule --name TestRule --schedule-expression 'cron(0 0/4 * * ? *)'
```

若要每五分鐘執行一次規則，請使用下列 cron 表達式。

```
aws events put-rule --name TestRule --schedule-expression 'cron(0/5 * * * ? *)'
```

使用 cron 運算式之 EventBridge 規則的最佳解析度為一分鐘。您的排程規則會在此一分鐘內運行，但不會精確地在第 0 秒時觸發。

因為 EventBridge 目標服務是分散式的，所以從排定的規則執行到目標服務對目標資源執行動作的時間之間，可能會有幾秒鐘的延遲。

我的規則與 AWS 全域服務 API 呼叫相符，但未執行

AWS 全球服務；例如 IAM 和 Amazon Route 53 僅在美國東部 (維吉尼亞北部) 區域提供，因此來自全球服務的 AWS API 呼叫事件僅在該區域提供。如需詳細資訊，請參閱 來自 AWS 服務的事件。

規則執行時，會忽略與我的規則關聯的 IAM 角色

EventBridge 只會將 IAM 角色用於將事件傳送至 Kinesis 串流的規則。針對調用 Lambda 函數或 Amazon SNS 主題的規則，您需要提供資源型許可。

確保您的區域 AWS STS 端點已啟用， EventBridge 以便在擔任您提供的 IAM 角色時可以使用它們。如需詳細資訊，請參閱 IAM 使用者指南AWS STS 中的在 AWS 區域中啟用和停用。

我的規則有一個應該匹配資源的事件模式，但沒有事件匹配

大多數服務 AWS 將冒號 (:) 或斜杠 (/) 視為 Amazon 資源名稱 (ARN) 中的相同字符。 ，但在事件模式和規則中 EventBridge 使用完全匹配。在建立事件模式時，請務必使用正確的 ARN 字元，使這些字元符合要匹配的事件中的 ARN 語法。

某些事件 (例如來自 CloudTrail的 AWS API 呼叫事件) 在資源欄位中沒有任何項目。

我的事件交付到目標時發生延遲

EventBridge 嘗試將事件傳遞至目標最多 24 小時，但在目標資源受到限制的情況下除外。第一次嘗試會在事件送達事件串流後立即執行。如果目標服務發生問題， EventBridge會自動重新排程另一個傳送。如果自事件到達以來已過 24 小時，則 EventBridge 會停止嘗試傳送事件並將`FailedInvocations`量度發佈於中 CloudWatch。建議您設定 DLQ 來儲存無法成功交付至目標的事件。如需更多資訊，請參閱事件重試政策和使用無效字母佇列

部分事件從未交付至我的目標

如果 EventBridge 規則的目標受到長時間限制，則 EventBridge 可能不會重試傳遞。例如，如果未佈建目標來處理內送事件流量，且目標服務正在限制代表您發 EventBridge 出的要求，則 EventBridge 可能不會重試傳遞。

在回應一個事件時，規則的運行次數超過一次

在極少數情況下，單一事件或排程時間可運行相同的規則一次以上，或者特定觸發的規則可多次調用相同的目標。

防止無限迴圈

在中 EventBridge，可以建立導致無限迴圈的規則，其中規則會重複執行。如果您有導致無限迴圈的規則，請將其重新寫入，讓規則採取的動作不符合相同的規則。

例如，如果規則會偵測到 Amazon S3 儲存貯體上的 ACL 已變更，然後執行軟體來將它們變更為新狀態的規則會導致無限迴圈。解決此問題的一種方法是重寫規則，讓其僅匹配處於不良狀態的 ACL。

無限循環可能會快速引發較預期還高的費用。我們建議您使用預測費用，也就是在費用超過您指定的限制時提醒您。如需詳細資訊，請參閱在符合預算的情形下管理成本。

我的事件不會傳送到目標 Amazon SQS 佇列

如果您的 Amazon SQS 佇列已加密，則必須建立客戶管理的 KMS 金鑰，並在 KMS 金鑰政策中包含下列許可區段。如需詳細資訊，請參閱設定 AWS KMS 權限。

```
{
  "Sid": "Allow EventBridge to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
}
```

我的規則運行，但我沒看到任何訊息發佈到我的 Amazon SNS 主題

案例 1

您需要許可才能將訊息發佈到您的 Amazon SNS 主題中。使用以下命令 AWS CLI，將 us-east-1 替換為您的區域並使用您的主題 ARN。

```
aws sns get-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic"
```

若要擁有正確的許可，您的策略屬性類似於以下內容。

```
"{\"Version\":\"2012-10-17\",
\"Id\":\"__default_policy_ID\",
\"Statement\":[{\"Sid\":\"__default_statement_ID\",
\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":\"*\"},
\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",
\"SNS:DeleteTopic\",
\"SNS:GetTopicAttributes\",
\"SNS:Publish\",
\"SNS:RemovePermission\",
\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\",
\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":\"123456789012\"}}},{\"Sid\":
\"Allow_Publish_Events\",
```

```
\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}]}"
```

如果您在原則中看不到具有 Publish 許可的 events.amazonaws.com 情況下，請先複製目前的政策，然後將下列陳述式新增至陳述式清單中。

```
{\"Sid\":\"Allow_Publish_Events\",
\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}
```

然後使用 AWS CLI，使用下列命令來設定主題屬性。

```
aws sns set-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic" --attribute-name Policy --attribute-
value NEW_POLICY_STRING
```

> **Note**
>
> 如果原則不正確，您也可以在 EventBridge 主控台中編輯規則，方法是移除規則，然後再將其新增回規則。 EventBridge 在目標上設定正確的權限。

案例 2

如果您的 SNS 主題已加密，您必須在 KMS 金鑰政策中包含下列區段。

```
{
  "Sid": "Allow EventBridge to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
```

```
}
```

EventBridge即使刪除了與 Amazon SNS 主題相關聯的規則，我的 Amazon SNS 主題仍然具有許可

當您以 Amazon SNS 做為目標建立規則時，請代表您 EventBridge 將權限新增至您的 Amazon SNS 主題。如果您在建立規則後不久刪除規則， EventBridge 可能不會從 Amazon SNS 主題中移除該許可。如果發生這種情況，您可以透過使用 `aws sns set-topic-attributes` 命令從該主題移除許可。如需有關傳送事件之資源型許可的詳細資訊，請參閱 將以資源為基礎的政策用於 Amazon EventBridge。

我可以使用哪些 IAM 條件金鑰 EventBridge？

EventBridge 支援 AWS寬範圍的條件金鑰 (請參閱《IAM 使用者指南》中的 IAM 和 AWS STS 條件內容金鑰)，以及中列出的金鑰使用 IAM 政策條件進行精細定義存取控制。

如何判斷 EventBridge 規則何時被破壞？

您可以使用以下警報在 EventBridge 規則中斷時通知您。

建立警示以在規則損壞時發出提醒

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 選擇建立警示 。在「依類別的CloudWatch 測量結果」窗格中，選擇事件測量結果
3. 在量度清單中，選取FailedInvocations。
4. 在圖形上方，選擇統計數據、總和。
5. 在期間中選擇一個值，例如 5 分鐘。選擇下一步。
6. 在警示臨界值下，為名稱輸入警示的唯一名稱，例如myFailedRules。在描述中輸入警示的描述，例如規則不會將事件交付至目標。
7. 在是 中選擇 >= 和 1。在 for 中輸入 10。
8. 在動作下的 每當此警示，選擇狀態為警示。
9. 在傳送通知至中選取現有 Amazon SNS 主題或建立新的主題。若要建立新的主題，請選擇新清單。輸入新 Amazon SNS 主題的名稱，例如：myFailedRules。
10. 在 Email list (電子郵件清單) 中輸入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM (警示) 狀態時，這些電子郵件地址將會收到通知。

11. 選擇建立警示。

Amazon EventBridge 配額

EventBridge 的大部分方面都有配額。

主題

> **Note**
>
> 如需 EventBridge 排程器的配額清單，請參閱《EventBridge 排程器使用者指南》中的EventBridge 接排程器配額。

EventBridge 配額

EventBridge 具有下列配額：

Service Quotas 主控台提供了有關 EventBridge 配額的資訊。除了檢視預設配額之外，您還可以使用 Service Quotas 主控台來請求增加配額以取得可調整配額。

名稱	預設	可調整	描述
API 目的地	每個受支援的區域：3,000 個	是	每個區域的每個帳戶的 API 目的地數上限
連線	每個受支援的區域：3,000 個	是	每個區域每個帳戶的連線數量上限。
建立每秒交易的端點限流限制	每個支援的區域：每秒 5 個	否	CreateEndpoint API 請求的每秒交易數量上限。其他請求會受到限流。

名稱	預設	可調整	描述
刪除每秒交易的端點限流限制	每個支援的區域：每秒 5 個	否	DeleteSecret API 請求的每秒交易數量上限。其他請求會受到限流。
端點	每個受支援的區域：100	是	每個區域每個帳戶的端點數量上限。
事件匯流排政策大小	每個受支援的區域：10,240	是	政策大小上限 (以字元為單位)。當您每次授與存取另一個帳戶時，此政策大小都會提高。您可以使用 DescribeEventBus API 查看目前的政策及其大小。
事件匯流排	每個受支援的區域：100	是	每個帳戶的最大事件匯流排。
事件模式大小	每個受支援的區域：2048 個	是	事件模式的大小上限，以字元為單位。

名稱	預設	可調整	描述
調用限流每秒交易的限制	us-east-1：每秒 18,750 個 us-east-2：每秒 4,500 個 us-west-1：每秒 2,250 個 us-west-2：每秒 18,750 個 af-south-1：每秒 750 個 ap-northeast-1：每秒 2,250 個 ap-northeast-3：每秒 750 個 ap-southeast-1：每秒 2,250 個 ap-southeast-2：每秒 2,250 個 ap-southeast-3：每秒 750 個 eu-central-1：每秒 4,500 個 eu-south-1：每秒 750 個	是	調用是符合規則並被傳送到規則目標的事件。達到上限之後，就會限流呼叫，也就是說，雖然仍會繼續呼叫，但是會延遲一些。

名稱	預設	可調整	描述
	eu-west-1：每秒 18,750 個 eu-west-2：每秒 2,250 個 每個其他支援的區域：每秒 1,100 個		
規則數目	af-south-1：750 eu-south-1: 100 每個其他支援的區域：300	是	每個事件匯流排帳戶可擁有的規則數目上限

名稱	預設	可調整	描述
PutEvents 限流每秒交易的限制	us-east-1：每秒 10,000 個 us-east-2：每秒 2,400 個 us-west-1：每秒 1,200 個 us-west-2：每秒 10,000 個 af-south-1：每秒 400 個 ap-northeast-1：每秒 1,200 個 ap-northeast-3：每秒 400 個 ap-southeast-1：每秒 1,200 個 ap-southeast-2：每秒 1,200 個 ap-southeast-3：每秒 400 個 eu-central-1：每秒 2,400 個 eu-south-1：每秒 400 個	是	每秒 PutEvents API 請求數上限。其他請求會受到限流。

名稱	預設	可調整	描述
	eu-west-1：每秒 10,000 個 eu-west-2：每秒 1,200 個 每個其他支援的區域：每秒 600 個		
每個 API 目標的調用速率	每個支援的區域：每秒 300 個	是	每個區域每個帳戶傳送至每個 API 目標端點的每秒調用數上限。一旦滿足配額，之後對該 API 端點的調用就會被限制。調用仍然會發生，但會延遲。
每個規則的目標	每個受支援的區域：5	否	可以與這些規則建立關聯的目標數目上限
限流每秒交易的限制	每個支援的區域：每秒 50 個	是	除 PutEvents 之外的所有 EventBridge API 操作每秒請求數上限。其他請求會受到限流。
UpdateEndpoint 限流每秒交易的限制	每個支援的區域：每秒 5 個	否	UpdateEndpoint API 請求的每秒交易數量上限。其他請求會受到限流。

此外，EventBridge 具有下列配額，這些配額不是透過「Service Quotas」主控台管理。

名稱	預設	描述
事件匯流排	每個受支援的區域：100	每個帳戶的最大事件匯流排。

名稱	預設	描述
事件匯流排政策大小	每個受支援的區域：10240	政策大小上限 (以字元為單位)。當您每次授與存取另一個帳戶時，此政策大小都會提高。您可以使用 DescribeEventBus API 查看目前的政策及其大小。
事件模式大小	每個受支援的區域：2048	事件模式的大小上限，以字元為單位。 這些字元最多可調整 4096 個字元。如果您需要更高的最大限制，請聯絡支援人員。
包含萬用字元的規則	每個支援的區域：每個事件匯流排 30 個規則	每個帳戶每個事件匯流排可包含包含萬用字元的事件篩選器的規則數目上限。此配額無法變更。 如需在事件模式中使用萬用字元的詳細資訊，請參閱 ???。
結構描述探索層	每個受支援的區域：255 層	結構描述探索的層級數目上限會推斷巢狀的事件。超過 255 個層級的任何事件都會被忽略。

按地區劃分的 PutPartnerEvents 配額

如果您需要更高上限，請聯絡支援部。

區域	每秒交易數
• AWS GovCloud (美國西部) • AWS GovCloud (US-East) • 美國東部 (維吉尼亞北部) • 美國東部 (俄亥俄) • 美國西部 (加利佛尼亞北部) • 美國西部 (奧勒岡) • 非洲 (開普敦) • 亞太區域 (香港) • 亞太區域 (孟買)	在所有區域中，PutPartNevets 的軟式限制為每秒 1,400 個輸送量要求，以及每秒 3,600 個突發要求。

區域	每秒交易數
• 亞太區域 (大阪) • 亞太區域 (首爾) • 亞太區域 (新加坡) • 亞太區域 (雪梨) • 亞太區域 (東京) • 加拿大 (中部) • 歐洲 (法蘭克福) • 歐洲 (愛爾蘭) • 歐洲 (倫敦) • 歐洲 (米蘭) • 歐洲 (巴黎) • 歐洲 (斯德哥爾摩) • 歐洲 (米蘭) • 南美洲 (聖保羅) • 中國 (寧夏) • 中國 (北京)	

EventBridge 結構描述登錄檔配額

EventBridge 結構描述登錄檔具有下列配額：

Service Quotas 主控台提供了有關 EventBridge 配額的資訊。除了檢視預設配額之外，您還可以使用 Service Quotas 主控台來請求增加配額以取得可調整配額。

名稱	預設	可調整	描述
DiscoveredSchemas	每個受支援的區域：200	是	您可以在目前區域中建立之探索之綱要登錄的結構描述數目上限

名稱	預設	可調整	描述
探索工具	每個受支援的區域：10	是	您可在目前區域中建立的探索工具上限數量。
登錄檔	每個受支援的區域：10	是	您可在目前區域中建立的登錄檔上限數量。
SchemaVersions	每個受支援的區域：100	是	您可在目前區域中建立的每結構描述版本上限數量。
結構描述	每個受支援的區域：100	是	您可在目前區域中建立的每登錄檔結構描述上限數量。(探索到的結構描述登錄檔除外)

EventBridge 管道配額

EventBridge 管道具有下列配額：如果您需要更高上限，請聯絡支援部。

資源	區域	預設值限制
每個帳戶的並發管道執行	• AWS GovCloud (美國西部) • AWS GovCloud (美國東部) • 中國 (寧夏) • 中國 (北京) • 亞太區域 (大阪) • 非洲 (開普敦) • 歐洲 (米蘭) • 美國東部 (俄亥俄) • 歐洲 (法蘭克福) • 美國西部 (加利佛尼亞北部)	1000

資源	區域	預設值限制
	• 歐洲 (倫敦) • 亞太區域 (雪梨) • 亞太區域 (東京) • 亞太區域 (新加坡) • 加拿大 (中部) • 歐洲 (巴黎) • 歐洲 (斯德哥爾摩) • 南美洲 (聖保羅) • 亞太區域 (首爾) • 亞太區域 (孟買) • 亞太區域 (香港) • 中東 (巴林) • 中國 (寧夏) • 中國 (北京) • 亞太區域 (大阪) • 非洲 (開普敦) • 歐洲 (米蘭)	
每個帳戶的並發管道執行	• 美國東部 (維吉尼亞北部) • 美國西部 (奧勒岡) • 歐洲 (愛爾蘭)	3000
每個帳戶的管道	全部	1000

Amazon EventBridge 標籤

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。在中 EventBridge，您可以將標籤指派給規則匯流排和事件匯流排。每個資源的上限為 50 個標籤。

您可以使用標籤來識別和組織資 AWS 源。許多 AWS 服務都支援標記，因此您可以將相同標籤指派給來自不同服務的資源，以指出資源是相關的。例如，您可以將相同的標籤指派給指派給 EC2 執行個體的 EventBridge 規則。

每個標籤有兩個部分：

- 標籤鍵，例如：`CostCenter`、`Environment` 或 `Project`。
  - 標籤鍵會區分大小寫。
  - 最大標籤索引鍵長度為 128 個 UTF-8 形式的 Unicode 字元。
  - 每個資源的每個標籤索引鍵都必須是唯一的。
  - 允許的字元包括可用 UTF-8 表示的英文字母、數字、空格，還有以下特殊字元：. : + = @ _ / - (連字號)。
  - 標籤禁止使用`aws:`前綴，因為它保留供 AWS 使用。您不可編輯或刪除具此字首的標籤金鑰或值。具此字首的標籤，不算在受資源限制的標籤計數內。
- 選用標籤值欄位，例如：`111122223333` 或 `Production`。
  - 每個標籤索引鍵只能有一個值。
  - 標籤值區分大小寫。
  - 忽略標籤值基本上等同於使用空字串。
  - 最大標籤值長度為 256 個 UTF-8 形式的 Unicode 字元。
  - 允許的字元包括可用 UTF-8 表示的英文字母、數字、空格，還有以下特殊字元：. : + = @ _ / - (連字號)。

> **Tip**
>
> 做為最佳實務，請決定大寫標籤的策略，並一致地在所有資源類型中實作該策略。例如，決定要使用 `Costcenter`、`costcenter` 還是 `CostCenter`，然後針對所有標籤使用相同的慣例。

您可以使用 EventBridge 主控台、 EventBridge API 或新 AWS CLI 增、編輯或刪除標籤。如需詳細資訊，請參閱下列內容：

- TagResourceUntagResource、和ListTagsForResource在 Amazon EventBridge API 參考
- 標記資源，無標記資源，並在參考 list-tags-for-resourceAWS CLI
- 《Resource Groups 使用者指南》中的使用標籤編輯器

文件歷史記錄

下表說明從 2018 年 7 月開始，每一版《Amazon EventBridge使用者手冊》的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

變更	描述	版本日期
從事件匯流排和規則產生 AWS CloudFormation 範本。	您現在可以從現有的 Amazon EventBridge 事件匯流排和規則產生 AWS CloudFormation 範本。 • 從 Amazon EventBridge 事件匯流排產生 AWS CloudFormation 範本	2022 年 11 月 18 日
推出 EventBridge 管道文件。	您現在可以建立管道，透過選用的篩選和擴充功能將來源連接到目標。 • 管道	2022 年 12 月 1 日
從事件匯流排和規則產生 AWS CloudFormation 範本。	您現在可以從現有的 Amazon EventBridge 事件匯流排和規則產生 AWS CloudFormation 範本。 • 從 Amazon EventBridge 事件匯流排產生 AWS CloudFormation 範本	2022 年 11 月 18 日
添加了 AmazonEventBridgePipesFullAccess 政策。	提供對 Amazon EventBridge Pipes 的完整存取。 • 適用於 EventBridge 管道的受管政策	2022 年 12 月 1 日
添加了 AmazonEventBridgePipesReadOnlyAccess 政策。	提供對 Amazon EventBridge Pipes 的唯讀存取權限。 • 適用於 EventBridge 管道的受管政策	2022 年 12 月 1 日
添加了 AmazonEve	提供對 Amazon EventBridge Pipes 的唯讀和操作員 (也就是停止和開始執行管道的功能) 存取權。	2022 年 12 月 1 日

變更	描述	版本日期
ntBridgeP ipesOpera torAccess政策。	• 適用於 EventBridge 管道的受管政策	
已更新 CloudWatc hEventsFu llAccess 政策。	已更新以匹配 `AmazonEventBridgeF ullAccess` 。 • AmazonEventBridgeFullAccess 政策	2022 年 12 月 1 日
已更新 CloudWatc hEventsRe adOnlyAccess 政策。	已更新以匹配 `AmazonEventBridgeR eadOnlyAccess` 。 • AmazonEventBridgeReadOnlyAccess 政策	2022 年 12 月 1 日
更新了事件模式中的內容過濾。	您現在可以使用 `suffix`、`equals-ignore-case` 、和 `$or`篩選選項來建立事件模式。 • Amazon EventBridge 事件模式中的內容過濾	2022 年 11 月 14 日
更新了 AmazonEve ntBridgeF ullAccess 政策	已新增使用 EventBridge 結構描述登錄和 EventBridge 排程器所需的權限。 • AmazonEventBridgeFullAccess 政策	2022 年 11 月 10 日
更新了 AmazonEve ntBridgeR eadOnlyAccess 政策	您現在可以檢視 Amazon EventBridge 架構登錄和 Amazon EventBridge 排程器資訊。 • AmazonEventBridgeReadOnlyAccess 政策	2022 年 11 月 10 日
更新了事件模式中的內容過濾。	您現在可以使用 `suffix`、`equals-ignore-case` 、和 `$or` 篩選選項來建立事件模式。 • Amazon EventBridge 事件模式中的內容過濾	2022 年 11 月 14 日

變更	描述	版本日期
更新了 AmazonEventBridgeFullAccess 政策	已新增使用 EventBridge 結構描述登錄和 EventBridge 排程器所需的權限。 • AmazonEventBridgeFullAccess 政策	2022 年 11 月 10 日
更新了 AmazonEventBridgeReadOnlyAccess 政策	您現在可以檢視 Amazon EventBridge 架構登錄和 Amazon EventBridge 排程器資訊。 • AmazonEventBridgeReadOnlyAccess 政策	2022 年 11 月 10 日
更新了 AmazonEventBridgeReadOnlyAccess 政策	您現在可以檢視端點資訊。 • AmazonEventBridgeReadOnlyAccess 政策	2022 年 4 月 7 日
已新增對全域端點的支援。	Amazon EventBridge 現在支援全域端點的使用，來讓您應用程式的區域錯誤處於可接受範圍並無需增加額外費用。要進一步了解，請參閱下列項目： • 透過全域端點和事件複寫讓應用程式具有區域容錯能力 • `CreateEndpoint`	2022 年 4 月 7 日
增加了對存檔和事件重播的支持。	Amazon EventBridge 現在支援使用存檔來存放事件，並支援事件重播以重播存檔中的事件。要進一步了解，請參閱下列項目： • 封存 Amazon EventBridge 事件. • `CreateArchive` • `StartReplay`	2020 年 11 月 5 日

變更	描述	版本日期
已新增對無效字母佇列和目標重試政策的支援。	Amazon EventBridge 現在支援使用無效字母佇列，以及為目標定義重試政策。要進一步了解，請參閱下列項目： • 事件重試政策和使用無效字母佇列. • PutTargets	2023 年 10 月 12 日
新增了對 JSONSchema 草圖 4 結構描述的支援。	Amazon EventBridge 現在支持 JSOSchema 草案 4 格式的架構。您現在也可以使用 EventBridge API 匯出結構描述。要進一步了解，請參閱下列項目： • Amazon EventBridge 模式 • 在 EventBridge 架構登錄檔 API 參考中的 Export	2020 年 9 月 28 日
為 EventBridge 架構登錄指定的以資源為基礎的政策。	Amazon EventBridge 架構登錄檔現在支援資源類型政策。如需詳細資訊，請參閱下列內容。 • Amazon EventBridge 結構的資源型政策 • 在 EventBridge 架構登錄檔 API 參考中的 Policy • AWS CloudFormation 使用者指南中的登錄策略資源類型	2020 年 4 月 30 日
事件匯流排標籤	此版本允許您建立和管理事件匯流排的標籤。您可以在建立事件匯流排時新增標籤，並呼叫相關 API 來新增或管理現有標籤。如需詳細資訊，請參閱下列內容。 • Amazon EventBridge 標籤 • 標籤類型政策 • TagResource • UntagResource • ListTagsForResource	2020 年 2 月 24 日

變更	描述	版本日期
增加的服務配額	Amazon EventBridge 現在已為 `PutEvents` 和調用增加配額。配額因區域而異，必要時可增加。	2020 年 2 月 11 日
新增了關於轉換目標輸入的新主題，並新增應用程式自動擴展事件的連結。	改善了輸入傳換器的說明文件。 • Amazon EventBridge 輸入轉換 • 使用輸入轉換器從事件中擷取資料，並將該資料輸入至目標 • 教學課程：使用輸入轉換器以自訂 EventBridge 傳送至事件目標的內容 新增應用程式自動擴展事件的連結。 • 應用程式自動擴展事件和 EventBridge • 來自 AWS 服務的事件	2019 年 12 月 20 日
以內容為基礎的篩選		2019 年 12 月 19 日
新增 Amazon Augmented AI 事件範例的連結。	在 Amazon SageMaker 中新增了 Amazon Augmented AI 主題的連結，以提供 Amazon Augmented AI 的範例事件。如需詳細資訊，請參閱下列內容。 • 在 Amazon Augmented AI 中使用事件 • 來自 AWS 服務的事件	2019 年 12 月 13 日
新增 Amazon Chime 事件範例的連結。	已新增 Amazon Chime 主題的連結，以提供該服務的範例事件。如需詳細資訊，請參閱下列內容。 • 使用 EventBridge 自動化 Amazon Chime • 來自 AWS 服務的事件	2019 年 12 月 12 日

變更	描述	版本日期
Amazon EventBridge Schemas	您現在可以在 Amazon EventBridge 中管理結構描述並產生事件的程式碼繫結。如需詳細資訊，請參閱下列內容。 • Amazon EventBridge 模式 • EventBridge 結構描述 API 參考 • AWS CloudFormation 中的EventSchemas 資源類型參照	2019 年 12 月 1 日
事件匯流排的 AWS CloudFormation 支援	AWS CloudFormation 現在支援 EventBus 資源。它也支援 EventBusPolicy 和規則資源中的 EventBusName 參數。如需詳細資訊，請參閱 Amazon EventBridge 資源類型參考。	2019 年 10 月 7 日
新的服務	Amazon EventBridge 初始版本。	2019 年 7 月 11 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。