開發人員指南

AWS Step Functions

AWS Step Functions: 開發人員指南

Table of Contents

什麼是 AWS Step Functions？

AWS Step Functions 是一種無伺服器協調服務，可讓您與AWS Lambda功能和其他功能整合， AWS 服務 以建置關鍵業務應用程式。透過 Step Functions 的圖形化主控台，您可以將應用程式的工作流程視為一系列事件驅動的步驟。

Step Functions 以狀態機器和任務為基礎。在 Step Functions 中，工作流程稱為狀態機器，該狀態機器是一系列事件驅動的步驟。工作流程中的每個步驟稱為狀態。任務狀態代表另一個 AWS 服務 (例如) 執行的工作單位。 AWS Lambda任務狀態可以調用任何 AWS 服務 或 API。

使用 Step Functions 的內建控制項，您可以檢查工作流程中每個步驟的狀態，以確保應用程式依預期順序執行。根據您的使用案例，您可以讓 Step Functions 呼叫 AWS 服務 (例如 Lambda) 來執行工作。您可以建立處理和發佈機器學習模型的工作流程。您可以使用 Step Functions 控制 AWS 服務 AWS Glue，例如建立擷取、轉換和載入 (ETL) 工作流程。也可以為需要人為互動的應用程式建立長時間執行的自動化工作流程。

> **Tip**
>
> 若要透過一系列互動模組熟悉 Step Functions 的主要功能，請參閱工作坊。 AWS Step Functions或者，依照下列入門教學課程來建立信用卡申請工作流程，開始使用「Step Functions 數」。

主題

- AWS SDK 與最佳化整合
- 標準和快速工作流程
- 使用案例
- 服務整合
- 支援的 區域
- 這是您第一次使用 Step Functions 嗎？

AWS SDK 與最佳化整合

若要呼叫其他 AWS 服務，您可以使用 Step 函 AWS 式的 SDK 整合，也可以使用 Step 函數的最佳化整合之一。

- AWS SDK 集成使您可以直接從狀態機調用 200 多個 AWS 服務中的任何一個，從而使您可以訪問九千多個 API 操作。
- Step Functions 的最佳化整合已自訂，以簡化狀態機器中的使用。

標準和快速工作流程

Step Functions 有兩種工作流程類型。標準工作流程只執行一次工作流程，最長可執行一年。這表示「標準」工作流程中的每個步驟只會執行一次。但是，Express 工作流程具有 at-least-once 工作流程執行，最多可以執行五分鐘。這表示「快速工作流程」中的一或多個步驟可能會執行多次，而工作流程中的每個步驟至少執行一次。

執行是您執行工作流程以執行工作的執行個體。標準工作流程非常適合長時間執行的可稽核工作流程，因為它們會顯示執行歷程記錄和視覺化偵錯 Express 工作流程非常適合工作 high-event-rate 負載，例如串流資料處理和 IoT 資料擷取。

標準工作流程規

- 每秒 2,000 次執行速率
- 每秒 4,000 次狀態轉換率
- 按狀態轉換定價
- 顯示執行歷程記錄和視覺偵錯
- Support 所有服務集成和模式

快速工作流程規

- 每秒 10 萬次執行速率
- 近乎無限制的狀態轉換率
- 按執行次數和持續時間定價
- 將執行歷史發送到 Amazon CloudWatch
- 根據啟用的日誌級別顯示執行歷史記錄和視覺化調試
- Support 所有服務集成和大多數模式

如需標準和快速工作流程的詳細資訊，包括 Step Functions 定價，請參閱下列內容：

- 標準與快速工作流程

- AWS Step Functions 定價

使用案例

Step Functions 會管理應用程式的元件和邏輯，因此您可以撰寫較少的程式碼，並專注於快速建置和更新應用程式。本節說明使用 Step Functions 數的典型使用案例。

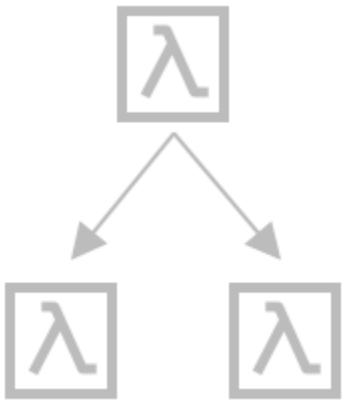
使用案例 #1: 函數協調

您可以建立以特定順序執行一組 Lambda 函數 (步驟) 的工作流程。一個 Lambda 函數的輸出傳遞到下一個 Lambda 函數的輸入。工作流程中的最後一個步驟會產生結果。使用 Step Functions，您可以查看工作流程中的每個步驟如何相互作用，以確保每個步驟都執行其預期功能。

如需說明如何使用一組函式建立狀態機器的教學課程，請參閱下列內容：

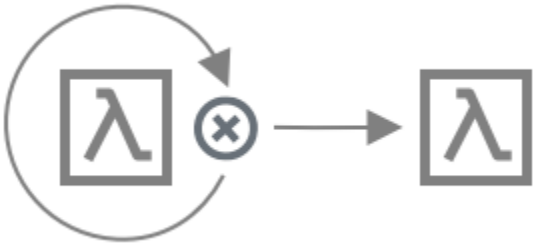
- AWS Step Functions 入門

使用案例 #2: 分支



客戶要求提高信用額度。使用`Choice`狀態，您可以讓 Step Functions 根據`Choice`狀態的輸入做出決定。如果請求超過客戶預先核准的信用額度，您可以讓 Step Functions 將客戶的請求傳送給經理以進行登出。如果請求小於客戶預先核准的信用額度，您可以讓「Step Functions」自動核准請求。

使用案例 #3: 錯誤處理



`Retry`

在此使用案例中，客戶要求使用者名稱。第一次，您的客戶的請求不成功。使用`Retry`語句，你可以讓 Step Functions 再次嘗試你的客戶的請求。第二次，您的客戶的請求成功。

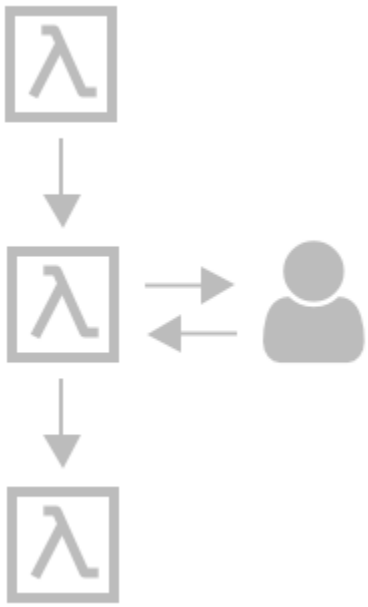
`Catch`

在類似的使用案例中，客戶要求不可用的使用者名稱。使用`Catch`語句，你有 Step Functions 建議一個可用的用戶名。如果您的客戶使用可用的使用者名稱，您可以讓 Step Functions 移至工作流程的下一個步驟，也就是傳送確認電子郵件。如果您的客戶沒有使用可用的使用者名稱，您可以讓 Step Functions 移至工作流程中的另一個步驟，即開始註冊程序。

如需`Retry`和`Catch`陳述式的詳細範例，請參閱下列內容：

- Step Functions 中的錯誤處理

使用案例 #4: 循環中的人

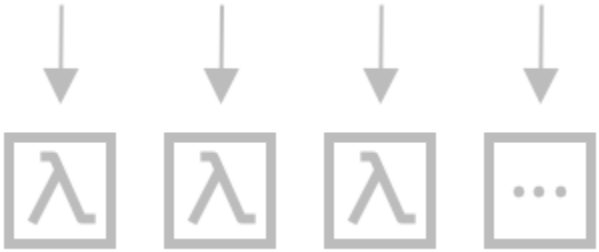


使用銀行應用程序，您的一位客戶向朋友匯款。您的客戶等待確認電子郵件。使用回調和任務令牌，您可以讓 Step Functions 告訴 Lambda 在客戶的朋友收到資金時向您的客戶發送資金並進行報告。Lambda 回報客戶的朋友收到款項後，您就可以讓 Step Functions 進入工作流程的下一個步驟，即傳送確認電子郵件給客戶。

要查看顯示帶有任務令牌的回調的示例項目，請參閱以下內容：

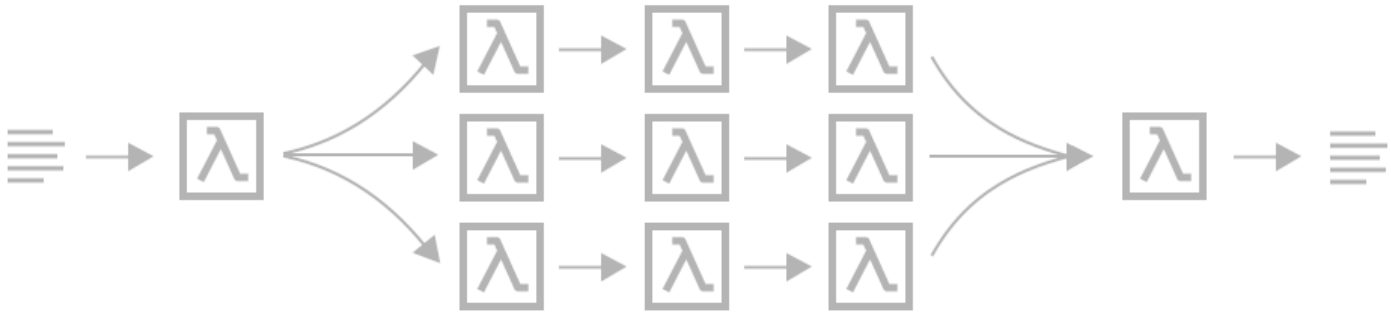
- 回呼模式範例 (Amazon SQS、Amazon SNS、Lambda)

使用案例 #5: 平行處理



客戶會將影片檔案轉換成五種不同的顯示解析度，讓觀眾可以在多個裝置上觀看影片。使用`Parallel`狀態，Step Functions 輸入視訊檔案，因此 Lambda 可以同時將其處理為五個顯示解析度。

使用案例 #6: 動態平行



一位客戶訂購三件商品，您需要準備每個項目以進行交貨。您可以檢查每個項目的可用性，收集每個項目，然後包裝每個項目以進行交付。使用Map狀態時，Step Functions 可讓 Lambda parallel 處理客戶的每個項目。一旦客戶的所有項目都包裝好交付，Step Functions 會進入工作流程的下一個步驟，即向客戶發送包含跟踪信息的確認電子郵件。

若要查看使用Map狀態顯示動態平行處理原則的範例專案，請參閱下列內容：

- 使用「地圖」狀態動態處理資料

服務整合

Step Functions 與多種 AWS 服務集成。若要將 Step Functions 與這些服務結合，請使用下列服務整合模式：

要求回應 (預設值)

- 呼叫服務，並讓 Step Functions 式在取得 HTTP 回應之後進入下一個狀態。

執行工作 (.sync)

- 呼叫服務，並讓 Step Functions 等待工作完成。

等待帶有任務令牌的回調（. waitForTask令牌）

- 使用任務令牌調用服務，並讓 Step Functions 等待，直到任務令牌返回與回調。

下表顯示 Step Functions 的可用服務整合和服務整合模式。

標準工作流程和 Express 工作流程支援相同的整合，但不支援相同的整合模式。

- 每個集成的優化集成模式支持都不同。
- Express Job 流程不支援執行工作 (.sync) 或等待回呼 (. waitForTask令牌 ）。
- 如需詳細資訊，請參閱 標準與快速工作流程。

Standard Workflows

支援的服務整合

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
最佳化整合	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
	AWS Lambda	✓		✓
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS SDK 整合	超過二百	✓		✓

Express Workflows

支援的服務整合

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
最佳化整合	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS SDK 整合	超過二百	✓		

支援的 區域

大多數 AWS 地區都支援 Step Functions。如需可使用「Step Functions」之區 AWS 域的完整清單，請參閱AWS 區域表。

這是您第一次使用 Step Functions 嗎？

如果這是您第一次使用 Step Functions，下列主題可協助您瞭解使用 Step Functions 數的不同部分，包括 Step Functions 如何與其他 AWS 服務結合：

- Step Functions 的教學課程
- Step Functions 的範例專案
- AWS Step Functions 數據科學開發套件

開始使用的先決條件 AWS Step Functions

第一次開始使AWS Step Functions用之前，請完成此頁面上列出的必要條件。

主題

步驟 1：註冊AWS 帳戶和 IAM 使用者

若要存取任何 AWS 服務，首先您必須建立 AWS 帳戶 帳戶。您可以使用您的 AWS 帳戶 帳戶來檢視您的活動和用量報告，以及管理身分驗證和存取。您只需為使用的產品和服務付費，而且您可以免費開始使AWS用。如需詳細資訊，請參閱 AWS 免費方案。

為了避免使用您AWS 帳戶根使用者的「Step Functions」動作，最佳做法是為每個需要管理 Step Functions 存取權的人員建立 IAM 使用者。

如果您已有AWS帳戶，請跳至下一個先決條件。

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

註冊後，請保護AWS 帳戶AWS 帳戶根使用者、啟用和建立系統管理使用者AWS IAM Identity Center，這樣您就不會將 root 使用者用於日常工作。

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 身分識別中心。

   如需指示，請參閱《AWS IAM Identity Center使用指南》AWS IAM Identity Center中的「啟用」。

2. 在 IAM 身分中心中，將管理存取權授與管理使用者。

   若要取得有關使用IAM Identity Center 目錄做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center者存取」。

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。

步驟 2：授予程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。

哪個使用者需要程式設計存取權？	到	By
		• 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

AWS Step Functions 入門

歡迎來到 Step Functions開始使用教程系列。

Step Functions 是一種無伺服器協同運作服務，可讓您將應用程式工作流程定義為一系列事件驅動步驟。工作流程中的每個步驟稱為狀態。您最常使用的狀態，例如任務,Choice,平行，以及Map，以定義您的工作流程。在Task狀態，您可以使用AWSStep Functions 支援和協調多個 SDK 整合AWS 服務在您的工作流程中。

主題

重要概念

本節向您介紹重要的 Step Functions 概念。在您開始運作之前，請檢閱下列關鍵業務概念。

術語	描述
工作流程	描述一系列步驟，而且通常與商務程序相符。
運作工作室	視覺化工作流程設計工具，可協助您更快地建立工作流程原型並建 如需詳細資訊，請參閱AWS Step Functions 流程工作室。
狀態	狀態機中的各個步驟，它們在狀態機中執行各種功能。如需詳細資訊，請參閱狀態。

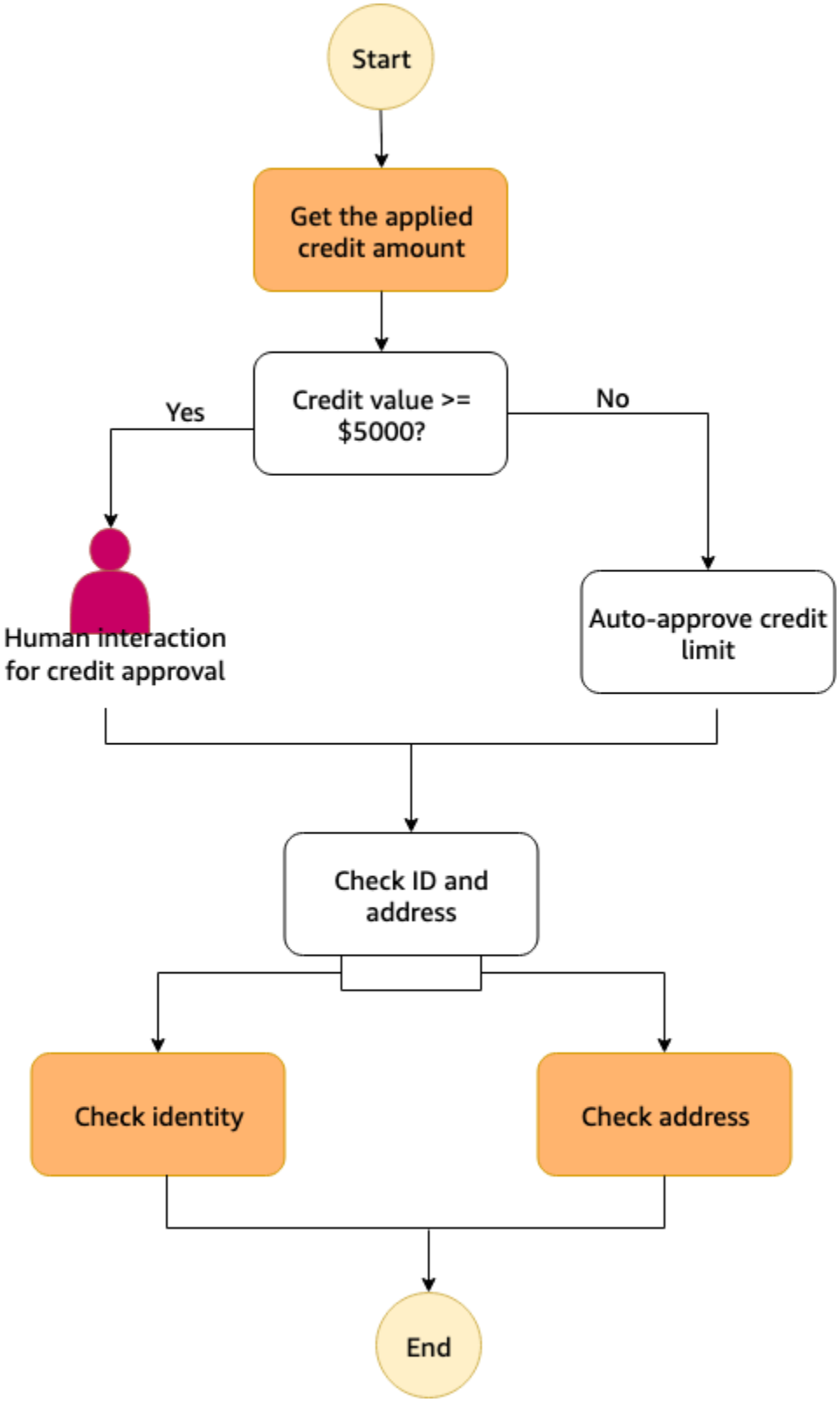
術語	描述
狀態機器	使用 JSON 文字來定義的工作流程，代表工作流程中的個別狀態或步驟，以及欄位，例如`StartAt`,`TimeoutSeconds` ，以及`Version`。如需詳細資訊，請參閱国家机结构。
Amazon States Language	一種基於 JSON 的結構化語言，用來定義您的狀態機。這是一種集合各州可以做工作 (`Task`狀態)，決定要轉換到下一個的狀態 (`Choice`狀態)，並停止執行並出現錯誤 (`Fail`狀態). 如需詳細資訊，請參閱Amazon States Language。
輸入和輸出配置	工作流程中的個別狀態會接收 JSON 資料做為輸入，通常會將 JSON 資料作為輸出傳遞至下一個狀態。Step Functions 提供了多個過濾器來控制狀態之間的輸入和輸出數據流。如需詳細資訊，請參閱Step Functions 中的輸入和輸出處理。
服務整合	Step Functions 直接整合AWS 服務，可讓您從工作流程呼叫每個服務的 API 動作。如需詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。
服務整合類型	Step Functions 提供下列服務整合類型： • 最佳化整合— 由 Step Functions 自訂，以提供工作流程的特殊功能。例如，Lambda 調用會將其 API 輸出從轉義的 JSON 字符串轉換為 JSON 對象。 • AWSSDK 整合— 行為與使用的標準 API 呼叫完全相同AWSSDK。你可以打電話給任何一個超過二百AWS 服務直接從您的狀態機上訪問超過九千個 API 操作。 如需詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。
服務整合模式	若要呼叫整合式AWS 服務在工作流程中，您可以使用 Step Functions 提供的下列服務整合模式之一： • 要求回應 (預設值)— 呼叫服務，並讓 Step Functions 式在取得 HTTP 回應後進入下一個狀態。 • 執行工作 (.sync)— 呼叫服務並讓 Step Functions 等待作業完成。 • 等待帶有任務令牌的回調 (.waitForTask令牌) -使用任務令牌調用服務，並讓 Step Functions 等待，直到任務令牌以回調返回。
執行	狀態機器執行是執行工作流程以執行工作的執行個體。如需詳細資訊，請參閱Step Functions 數中的執行。

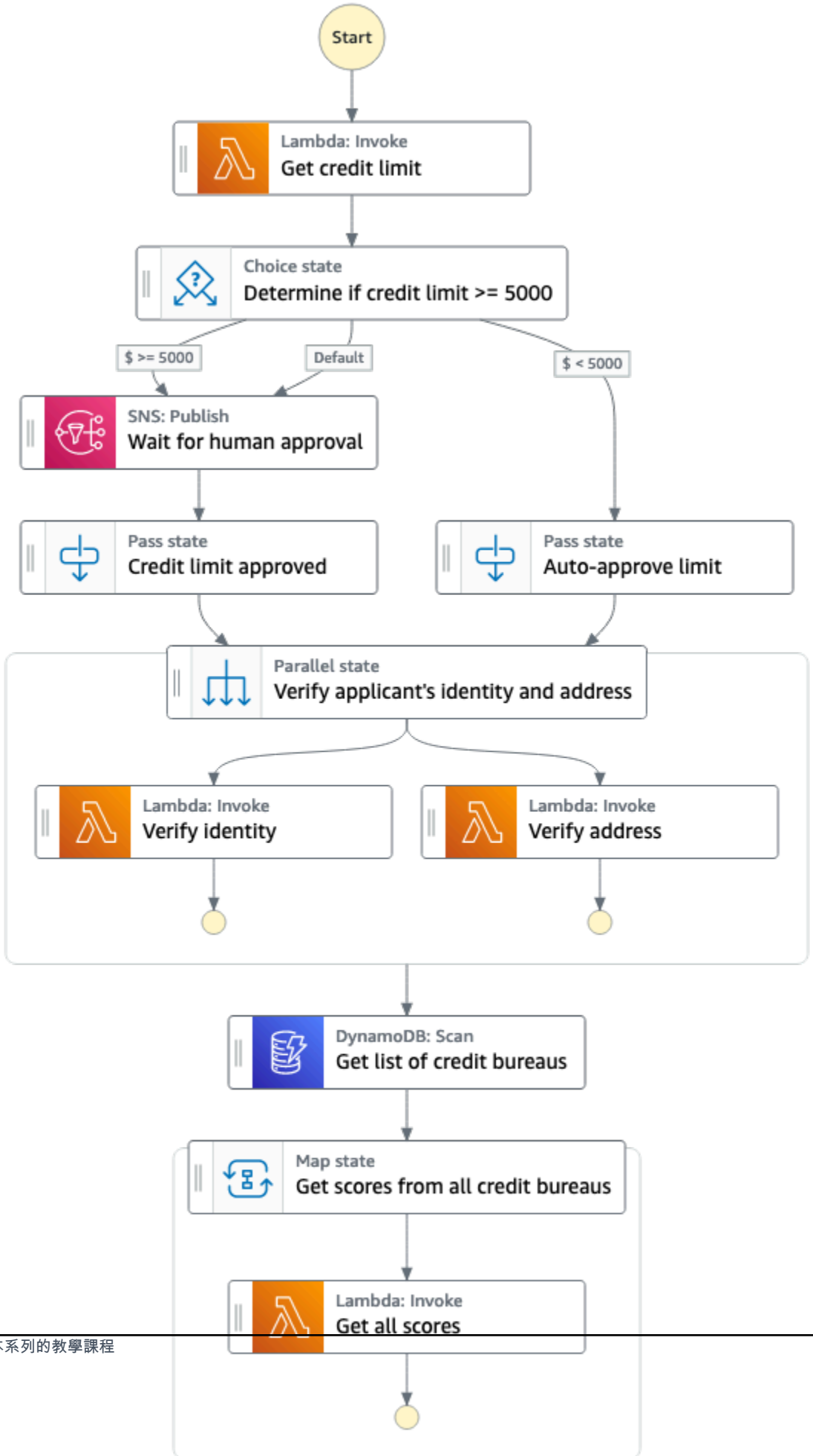
本系列的教學課程

該開始使用本章的自學課程將引導您逐步建立處理信用卡申請的基本工作流程。在這些教學課程中，您將學習如何在 Step Functions 中使用常用狀態。您將與其他工作流程整合AWS 服務，例如AWS Lambda和亞馬遜簡單通知服務。完成這些教學課程後，您將擁有一個模擬處理信用卡申請的簡單工作流程。

> **Note**
>
> 雖然這些開始使用自學課程說明信用卡申請工作流程，您可以使用「Step Functions」來建立多種類型的工作流程。例如，您可以建立資料處理、IT 自動化、機器學習、媒體處理或訂單處理的工作流程。

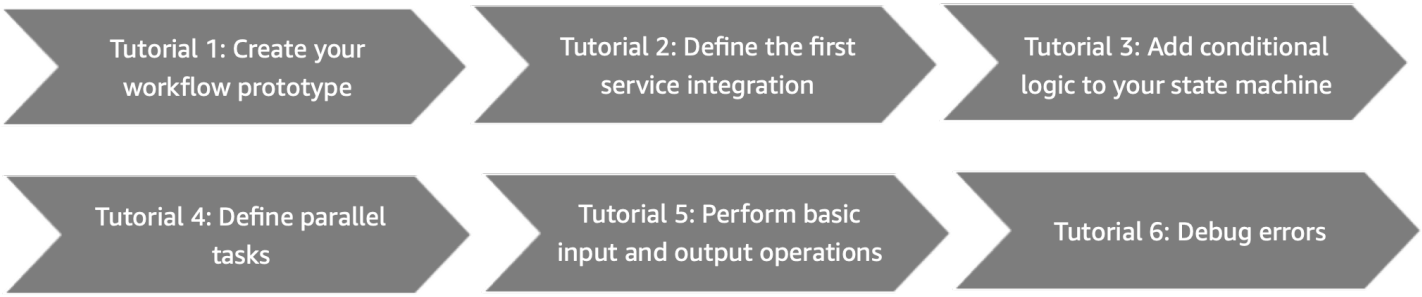
下列影像代表信用卡申請工作流程，以及使用 Step Functions 進行編排時的顯示方式。流程圖中的每個步驟都以「步 Step Functions」工作流程中的狀態表示。


Start
Get the applied credit amount
Credit value >= $5000?
Yes
No
Human interaction for credit approval
Auto-approve credit limit
Check ID and address
Check identity
Check address
End


Start
Lambda: Invoke
Get credit limit
Choice state
Determine if credit limit >= 5000
$ >= 5000
Default
$ < 5000
SNS: Publish
Wait for human approval
Pass state
Credit limit approved
Pass state
Auto-approve limit
Parallel state
Verify applicant's identity and address
Lambda: Invoke
Verify identity
Lambda: Invoke
Verify address
DynamoDB: Scan
Get list of credit bureaus
Map state
Get scores from all credit bureaus
Lambda: Invoke
Get all scores

> **Note**
>
> 我們建議您依序完成所有這些自學課程。完成完整的教學課程將教導您如何在生產工作流程中使用 Step Functions 的關鍵概念和功能。

下列藍圖顯示使用步驟函式工作流程 Studio 建立信用卡處理工作流程時所要執行的步驟。這些步驟會顯示為一系列自學課程，其中包含如何完成該步驟的指示。



在您開始運作之前，請務必完成先決條件。

教學 1：建立狀態機的原型

在本教學課程中，您會使用以建立信用卡處理工作流程的原型步驟函數工作流程工作室。您將從中選擇所需的 API 動作和狀態動作和流程分別選項卡，並使用工作流 Studio 的拖放功能來創建工作流原型。在後續的教學課程中，您將學習如何設定AWS 服務和步驟函數的狀態，您將在此工作流程中使用。

若要建立狀態機原型

1. 開啟連接器Step Functions 主控台並選擇建立狀態機。
2. 在選擇一個範本」對話方塊中，選取空白。
3. 選擇 Select (選取)。這會開啟工作流程工作室設計模式。
4. 在工作流程工作室中，從動作」頁籤中，拖曳AWS Lambda調用API 動作並將其放置到標記為空的狀態將第一個狀態拖到此。配置它，如下所示：
   - 在配置標籤，用於州名，輸入**`Get credit limit`**。
5. 從流程」索引標籤中，拖放選擇下面的狀態獲得信用限額狀態。重新命名選擇狀態至**`Credit applied >= 5000?`**。
6. 將下列狀態拖放為應用的信用額度大於等於 5000？狀態。

a. Amazon SNS 發布— 從動作」頁籤上，拖放Amazon SNS 發布API 動作。將此狀態重新命名為**Wait for human approval**。

b. 通行證狀態-從流程」頁籤上，拖放通行證狀態。將此分支重新命名為**Auto-approve limit**。

7. 拖放一個通行證下面的狀態等待人工批准狀態。重新命名通行證狀態至**Credit limit approved**。

8. 拖放一個平行之後的狀態選擇狀態如下：

a. 放下平行之後的狀態已核准信用額狀態。

b. 重新命名平行狀態至**Verify applicant's identity and address**。

c. 在這兩個分支平行狀態，拖放兩個AWS Lambda調用API 動作。

d. 將這些狀態重命名為**Verify identity**和**Verify address**分別。

e. 選擇合適的自動核准限制狀態和下一個狀態，可以選取核實申請人的身份及地址。

9. 拖曳一個掃描狀態並將其放在下面核實申請人的身份及地址狀態。重新命名掃描狀態至**Get list of credit bureaus**。

10. 拖放一個地圖之後的狀態取得信用機構清單狀態。配置地圖狀態如下：

a. 將其重新命名為**Get scores from all credit bureaus**。

b. 對於處理模式，也可以保留預設選取內聯。

c. 拖放AWS Lambda調用API 操作到標記為空狀態在此處放置狀態。

d. 重新命名AWS Lambda調用狀態至**Get all scores**。

11. 保持此視窗開啟，並繼續下一個教學課程，以進行進一步的動作。

後續步驟

在下一個教學課程中，您將學習如何整合所使用的 Lambda 函數獲得信用限額狀態。

教學課程 2：使用 Lambda 函數定義第一個服務整合

在本教學課程中，您將學習如何定義工作流程的第一個服務整合。您可以使用名為「取得信用額度」的Task狀態來叫用 Lambda 函數。在Task狀態中，您可以使用步驟函數支援的 AWS SDK 整合。

若要為您的工作流程定義第一個服務整合，請先建立 Lambda 函數。然後，更新您的工作流程，以指定與 Lambda 函數的服務整合。本教程中使用的 Lambda 函數返回一個隨機生成的整數，表示申請人已申請的信用限額。

主題

步驟 1：建立並測試 Lambda 函數

您可以在AWS Management Console或您最喜愛的編輯器中撰寫函式的程式碼。在下列步驟中，您會建立名為的 Node.js Lambda 函數`RandomNumberforCredit`。

> **Important**
>
> 請確定您在教學課程 1 中建立的工作流程原型與AWS 區域您將在本教學課程中建立的 Lambda 函數相同。

1. 在新索引標籤或視窗中，開啟 Lambda 主控台並建立標題**`RandomNumberforCredit`**為 Node.js 16.x Lambda 函數。如需使用主控台建立 Lambda 函數的詳細資訊，請參閱AWS Lambda開發人員指南中的主控台中的建立 Lambda 函數。
2. 在RandomNumberforCredit頁面上，選擇 index.mjs，並以下列程式碼取代「程式碼原始程式碼」區域中的現有程式碼。

```
export const handler = async function(event, context) {

    const credLimit = Math.floor(Math.random() * 10000);
    return (credLimit);

};
```

3. 從「函數概觀」區段中，複製 Lambda 函數的 Amazon 資源名稱，並將其儲存在文字檔中。指定 [取得信用額度] 狀態的服務整合時，您將需要函數 ARN。以下是 ARN 的示例：

```
arn:aws:lambda:us-east-2:123456789012:function:HelloWorld
```

4. 選擇部署，然後選擇測試以部署變更並查看 Lambda 函數的輸出。

步驟 2：更新工作流程 — 設定 [取得信用額度] 狀態

在 Step Functions 主控台中，您將更新工作流程，以指定與您在步驟 1 中建立的 RandomNumberforCredit Lambda 函數的服務整合。

1. 開啟包含您在教學課程 1 中建立的工作流程原型的「步驟函數」主控台視窗
2. 選擇「取得信用額度」狀態，然後在「組態」頁標中執行下列動作：
   a. 對於「整合」類型，請保留預設選取「最佳化」。

      使用 Step Functions，您可以與其他功能整合，AWS 服務並在工作流程中協調它們。如需服務整合及其類型的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。
   b. 對於函數名稱，從下拉式清單中選擇 RandomNumberforCreditLambda 函數。
   c. 保留其餘項目的預設選取項目。
3. 保持此視窗開啟，並繼續下一個教學課程，以進行進一步的動作。

> **Note**
>
> 在本教學課程中，您學習瞭如何與工作流程中某個Task狀態內的 Lambda 函數整合。您也可以指定服務名稱和 API 呼叫，在Task狀態中使用其他支援的 AWS SDK 整合，如下列語法所示：
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> 如需詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

下一步驟

在下一個教學課程中，您將在工作流程中實作條件式邏輯。在步驟函數狀態機條件邏輯的行為類似於在大多數常見的編程語言的 if-else 語句。您將在工作流程中使用條件式邏輯，根據條件資訊判斷執行路徑。

教學課程 3：在工作流程中實作 if-else 條件

您可以使用狀態`Choice`，在工作流程中實作 if-else 條件。它會根據指定的條件評估為 true 或 false 來決定工作流程執行路徑。

在本教學課程中，您將新增條件式邏輯，以判斷教學課程 2 中使用的 `RandomNumberforCredit` Lambda 函數傳回的已套用信用金額是否超過特定的閾值限制。如果金額超過閾值限制，則應用程序需要人工互動以進行批准。否則，應用程式會自動核准，並進入下一個步驟。

您將通過暫停工作流執行，直到返回任務令牌來模仿人工互動步驟。為此，您將傳遞一個任務令牌到您將在本教程中使用的 AWS SDK 集成，這是亞馬遜簡單通知服務。工作流程執行將暫停，直到它通過`SendTaskSuccess`API 調用收到任務令牌返回為止。如需有關使用工作權杖的更多資訊，請參閱等候傳回任務字符的回呼。

因為您已經在工作流程原型中定義了人工核准和自動核准的步驟，因此在本教學中，您會先建立接收回呼 Token 的 Amazon SNS 主題。然後，您可以建立 Lambda 函數來實作回呼功能。最後，您可以透過新增這些AWS 服務整合的詳細資訊來更新工作流程原型。

主題

步驟 1：建立接收回呼權杖的 Amazon SNS 主題

若要實作人工互動步驟，您將發佈到 Amazon 簡單通知服務主題，並將回呼任務 Token 傳遞給此主題。回呼工作將暫停工作流程執行，直到傳回包含有效負載的工作 Token 為止。

1. 開啟 Amazon SNS 主控台並建立標準主題類型。如需建立主題的相關資訊，請參閱 Amazon 簡單通知服務開發人員指南中的建立 Amazon SNS 主題。
2. 將主題名稱指定為**`TaskTokenTopic`**。
3. 確保複製主題 ARN 並將其保存在文本文件中。指定 [等待人工核准] 狀態的服務整合時，您需要主題 ARN。以下是 ARN 的範例主題：

```
arn:aws:sns:us-east-2:123456789012:TaskTokenTopic
```

4. 為主題建立以電子郵件為基礎的訂閱，然後確認您的訂閱。如需訂閱主題的相關資訊，請參閱 Amazon 簡單通知服務開發人員指南中的建立主題訂閱。

第 2 步：創建一個 Lambda 函數來處理回調

若要處理回呼功能，您需要定義 Lambda 函數，並將您在步驟 1 中建立的 Amazon SNS 主題新增為此函數的觸發器。當您使用任務權杖發佈到 Amazon SNS 主題時，會使用已發佈訊息的承載叫用 Lambda 函數。

- 步驟 2.1：創建 Lambda 函數來處理回調
- 步驟 2.2：將亞馬遜 SNS 主題新增為 Lambda 函數的觸發器
- 步驟 2.3：為 Lambda 函數 IAM 角色提供必要的許可

步驟 2.1：創建 Lambda 函數來處理回調

在此功能中，您將處理信用額度核准請求，並透過 SendTaskSuccessAPI 呼叫成功傳回要求的結果。此 Lambda 函數也會傳回從 Amazon SNS 主題收到的任務權杖。

為了簡單起見，用於人工互動步驟的 Lambda 函數會自動核准任何工作，並透過 SendTaskSuccess API 呼叫傳回任務 Token。您可以將 Lambda 函數命名為**callback-human-approval**。

1. 在新索引標籤或視窗中，開啟 Lambda 主控台並建立標題**callback-human-approval**為 Node.js 16.x Lambda 函數。如需使用主控台建立 Lambda 函數的詳細資訊，請參閱AWS Lambda開發人員指南中的主控台中的建立 Lambda 函數。
2. 在callback-human-approval頁面上，以下列程式碼取代 [程式碼原始碼] 區域中的現有程式碼。

```
// Sample Lambda function that will automatically approve any task whenever a
 message is published to an Amazon SNS topic by Step Functions.

console.log('Loading function');
const AWS = require('aws-sdk');
const resultMessage = "Successful";

exports.handler = async (event, context) => {
    const stepfunctions = new AWS.StepFunctions();

    let message = JSON.parse(event.Records[0].Sns.Message);
    let taskToken = message.TaskToken;
```

```
    console.log('Message received from SNS:', message);
    console.log('Task token: ', taskToken);

    // Return task token to Step Functions

    let params = {
        output: JSON.stringify(resultMessage),
        taskToken: taskToken
    };

    console.log('JSON Returned to Step Functions: ', params);
    let myResult = await stepfunctions.sendTaskSuccess(params).promise();
    console.log('State machine - callback completed..');

    return myResult;

};
```

3. 保持此視窗開啟，並執行下一節中的步驟，以進行進一步的動作。

步驟 2.2：將亞馬遜 SNS 主題新增為 Lambda 函數的觸發器

當您將在本教學課程步驟 1 中建立的 Amazon SNS 主題新增為您在本教學課程步驟 2.1 中建立的 Lambda 函數的觸發器時，每次您發佈到 Amazon SNS 主題時都會觸發 Lambda 函數。當您使用任務權杖發佈到 Amazon SNS 主題時，會使用已發佈訊息的承載叫用 Lambda 函數。如需設定 Lambda 函數觸發器的詳細資訊，請參閱開發AWS Lambda人員指南中的設定觸發程序。

1. 在 `callback-human-approval` Lambda 函數的 [函數概觀] 區段中，選擇 [新增觸發器]。
2. 從觸發器的下拉式清單中，選擇 SNS 做為觸發器。
3. 對於 SNS 主題，請開始輸入您在本教學課程的步驟 1 中建立的 Amazon SNS 主題名稱，然後從顯示的下拉式清單中選擇名稱。
4. 選擇 Add (新增)。
5. 保持此視窗開啟，並執行下一節中的步驟，以進行進一步的動作。

步驟 2.3：為 Lambda 函數 IAM 角色提供必要的許可

您必須提供 `callback-human-approval` Lambda 函數的許可，才能存取步驟函數，以便傳回任務權杖以及 `SendTaskSucess` API 呼叫。

1. 在callback-human-approval頁面上，選擇 [組態] 索引標籤，然後選擇 [權限]。
2. 在 [執行角色] 底下，選擇 [角色] 名稱以導覽至主AWS Identity and Access Management控台的 [角色] 頁面。
3. 若要新增必要的權限，請選擇 [新增權限]，然後選擇 [附加原則]。
4. 在搜尋方塊中，輸入，**`AWSStepFunctions`**然後按 Enter 鍵。
5. 選擇AWSStepFunctionsFullAccess然後向下捲動以選擇 [附加原則]。這會新增包含 `callback-human-approval` Lambda 函數角色必要權限的原則。

第 3 步：更新工作流-添加 if-else 條件邏輯在選擇狀態

在「步驟函數」主控台中，使用Choice狀態為工作流程定義條件式邏輯。如果 `RandomNumberforCredit` Lambda 函數傳回的輸出小於 5000，則會自動核准要求的點數。如果傳回的輸出大於或等於 5000，則工作流程執行會繼續進行信用額度核准的人工互動步驟。

在Choice狀態下，您可以使用比較運算子來比較輸入變數與特定值。您可以在啟動狀態機器執行時將輸入變數指定為執行輸入，或使用前一個步驟的輸出作為目前步驟的輸入。默認情況下，一個步驟的輸出存儲在一個名為的變量`Payload`。若要在Choice狀態中使用`Payload`變數的值進行比較，請使用下列程序所示的`$`語法。

有關資訊如何從一個狀態流向另一個狀態，以及如何在工作流程中指定輸入和輸出的資訊，請參閱教學課程 7：設定輸入和輸出和Step Functions 中的輸入和輸出處理。

> **Note**
>
> 如果狀Choice態使用狀態機器執行輸入中指定的輸入變數進行比較，請使用`$.variable_name`語法來執行比較。例如，若要比較變數，例如`myAge`，請使用語法`$.myAge`。

因為在這個步驟中，Choice狀態會從 [取得信用額度] 狀態接收輸入，所以您將使用Choice狀態設定的`$`語法。若要探索當您使用狀態組態中的`$.variable_name`語法來參照上述步驟的輸出時，Choice狀態機器執行的結果有何不同，請參閱教學課程 8 中的偵錯無效路徑選擇狀態錯誤章節。

若要使用狀態新增 if-else 條件邏輯 **`Choice`**

1. 開啟包含您在中建立之工作流程原型的「步驟函數」主控台視窗教學 1：建立狀態機的原型。

2. 選擇已套用的信用額度大於等於 5000？ state 並在「組態」索引標籤中，指定條件式邏輯，如下所示：

   a. 在「選擇規則」下，選擇「規則 #1」圖標中的「編輯」 圖示，以定義第一個選擇規則。

   b. 選擇 [新增條件]。

   c. 在「規則 #1 的條件」對話方塊中，對於「變數」，輸入**$**。

   d. 對於「運算子」，選擇小於。

   e. 對於「值」，請選擇「數字常數」，然後**5000**在「值」下拉式清單旁邊的欄位中輸入。

   f. 選擇 [儲存條件]。

   g. 對於「然後」下一個狀態為：下拉式清單，請選擇「自動核准限制」。

   h. 選擇「新增選擇規則」，然後在貸方金額大於或等於 5000 時，透過重複子步驟 2.b 到 2.f 來定義第二個選擇規則。對於「運算子」，請選擇「大於」或「等於」。

   i. 對於「然後」下一個狀態為：下拉式清單，請選擇「等待人工核准」。

   j. 在 [預設規則] 方塊中，選擇 [編輯] 圖示以定義預設選擇規則，然後從 [預設狀態] 下拉式清單中選擇 [等待人工核准]。如果沒有任何「選擇」狀態條件評估為 true 或 false，您可以定義「預設」規則，以指定要轉換到的下一個狀態。

3. 設定 [等待人工核准] 狀態，如下所示：

   a. 在「組態」索引標籤的「主題」中，開始輸入 Amazon SNS 主題的名稱 TaskTokenTopic，然後選擇下拉式清單中顯示的名稱。

   b. 對於「訊息」，從下拉式清單中選擇「輸入訊息」。在訊息欄位中，您可以指定要發佈到 Amazon SNS 主題的訊息。在本教學課程中，您會將工作 Token 發佈為訊息。

      工作 Token 可讓您暫停標準類型步驟函數工作流程，直到外部處理程序完成並傳回工作 Token 為止。當您透過指定[.waitForTaskToken服務整合模式將 T ask 狀態指定為回呼任務]時，會產生任務 Token，並在任務啟動時將其置於前後關聯物件中。上下文對象是一個內部 JSON 結構，可在執行期間使用，並包含有關您的狀態機器及其執行的信息。如需前後關聯物件的詳細資訊，請參閱[內容物件]。

   c. 在出現的方塊中，輸入以下內容作為訊息：

      ```
      {
        "TaskToken.$": "$$.Task.Token"
      }
      ```

   d. 選擇等待回呼核取方塊。

   e. 在出現的對話方塊中選擇「完成」。

4. 保持此視窗開啟，並繼續下一個教學課程，以進行進一步的動作。

後續步驟

在下一個教程中，您將學習如何並行執行多個任務。

教學課程 4：定義要平行執行的多個工作

到目前為止，您已經學會瞭如何以連續的方式執行工作流程。但是，您可以使用Parallel狀態並行執行兩個或多個步驟。狀Parallel態會導致解釋器同時執行每個分支。

Parallel狀態下的兩個分支都接收相同的輸入，但是每個分支都處理特定於它的輸入部分。Step 函數等待，直到每個分支完成執行，然後再繼續下一步。

在本教學課程中，您會使用「平行」狀態來同時檢查申請人的身分和地址。

主題

步驟 1：建立 Lambda 函數以執行所需的檢查

此信用卡應用程式工作流程會呼叫「平行」狀態內的兩個 Lambda 函數，以檢查申請人的身分和地址。這些檢查是使用「平行」狀態同時執行的。狀態機只有在兩個平行分支都完成執行之後才會完成執行。

若要建立檢查身分識別和檢查地址 Lambda 函數

1. 在新索引標籤或視窗中，開啟 Lambda 主控台，並建立兩個標題為check-identity和check-address的 Node.js 16.x Lambda 函數。如需使用主控台建立 Lambda 函數的詳細資訊，請參閱AWS Lambda開發人員指南中的主控台中的建立 Lambda 函數。
2. 開啟 [檢查識別功能] 頁面，並以下列程式碼取代 [程式碼原始碼] 區域中的現有程式碼：

```
const ssnRegex = /^\d{3}-?\d{2}-?\d{4}$/;
const emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;

class ValidationError extends Error {
    constructor(message) {
        super(message);
```

```
        this.name = "CustomValidationError";
    }
}

exports.handler = async (event) => {
    const {
        ssn,
        email
    } = event;
    console.log(`SSN: ${ssn} and email: ${email}`);

    const approved = ssnRegex.test(ssn) && emailRegex.test(email);

    if (!approved) {
        throw new ValidationError("Check Identity Validation Failed");
    }

    return {
        statusCode: 200,
        body: JSON.stringify({
            approved,
            message: `Identity validation ${approved ? 'passed' : 'failed'}`
        })
    }
};
```

3. 開啟 [檢查位址功能] 頁面，並以下列程式碼取代 [程式碼原始碼] 區域中的現有程式碼：

```
class ValidationError extends Error {
  constructor(message) {
      super(message);
      this.name = "CustomAddressValidationError";
  }
}

exports.handler = async event => {
  const {
      street,
      city,
      state,
      zip
  } = event;
  console.log(`Address information: ${street}, ${city}, ${state} - ${zip}`);
```

```
  const approved = [street, city, state, zip].every(i => i?.trim().length > 0);

  if (!approved) {
      throw new ValidationError("Check Address Validation Failed");
  }

  return {
      statusCode: 200,
      body: JSON.stringify({
          approved,
          message: `Address validation ${ approved ? 'passed' : 'failed'}`
      })
  }
};
```

4. 對於這兩個 Lambda 函數，請從函數概述部分複製各自的 Amazon 資源名稱 (ARN)，並將它們儲存在文字檔中。在為驗證申請人的身份和地址狀態指定服務集成時，您將需要功能 ARN。以下是 ARN 的示例：

```
arn:aws:lambda:us-east-2:123456789012:function:HelloWorld
```

步驟 2：更新工作流程 — 新增要執行的平行工作

在 Step Functions 主控台中，您將更新工作流程，以指定服務整合，以及您在步驟 1 中建立的檢查身分識別和檢查地址 Lambda 函數。

若要在工作流程中新增平行工作

1. 開啟包含您在中建立之工作流程原型的「步驟函數」主控台視窗教學 1：建立狀態機的原型。
2. 選擇 [驗證身分識別] 狀態，然後在 [組態] 索引標籤中執行下列動作：
   a. 對於「整合」類型，請保留預設選取「最佳化」。

      > **Note**
      >
      > 使用 Step Functions，您可以與其他功能整合，AWS 服務並在工作流程中協調它們。如需服務整合及其類型的詳細資訊，請參閱 AWS Step Functions 搭配其他服務使用

   b. 對於函數名稱，從下拉式清單中選擇檢查身分 Lambda 函數。
   c. 針對「承載」，選擇「輸入承載」，然後將範例承載取代為下列作為承載：

```
{
    "email": "janedoe@example.com",
    "ssn": "012-00-0000"
}
```

3. 選擇 [驗證位址] 狀態，然後在 [組態] 索引標籤中執行下列動作：

   a. 對於「整合」類型，請保留預設選取「最佳化」。

   b. 對於函數名稱，從下拉式清單中選擇檢查地址 Lambda 函數。

   c. 針對「承載」，選擇「輸入承載」，然後將範例承載取代為下列作為承載：

```
{
    "street": "123 Any St",
    "city": "Any Town",
    "state": "AT",
    "zip": "01000"
}
```

4. 選擇 下一步。

教學課程 5：同時迭代一系列項目

在上一個自學課程中，您學習瞭如何使用Parallel狀態並行執 parallel 不同的步驟分支。使用此狀Map態，您可以針對資料集中的每個項目執行一組工作流程步驟。Map狀態的反覆項目會 parallel 執行，因此可以快速處理資料集。

透過在工作流程中包含Map狀態，您可以使用以下兩種方式之一來執行工作，例如資料處理對映狀態處理模式：內嵌模式和分散式模式。若要設定Map狀態，您可以定義一個ItemProcessor，其中包含指定Map狀態處理模式及其定義的 JSON 物件。在本教學課程中，您會以預設的內嵌模式執行Map狀態，該模式最多支援 40 個並行反覆運算。當您以分散式模式執行Map狀態時，它最多支援 10,000 個 parallel 子工作流程執行。

當您的工作流程執行進入Map狀態時，它將遍歷狀態輸入中指定的 JSON 數組。對於每個陣列項目，其對應的版序會在包含Map狀態的工作流程前後關聯中執行。當所有迭代完成時，Map狀態將返回一個數組，其中包含由處理的每個項目的輸出ItemProcessor。

在本教程中，您將學習如何使用內聯模式下的Map狀態通過迭代一組信用機構來獲取申請人的信用評分。若要這麼做，您首先擷取 Amazon DynamoDB 表格中存放的所有信用機構名稱，然後使用Map州/省循環檢視信用機構清單，以擷取每個機構報告的申請人的信用評分。

主題

步驟 1：建立 DynamoDB 表格以儲存所有信用機構的名稱

在此步驟中，您可以**GetCreditBureau**使用 DynamoDB 主控台建立名為的資料表。此資料表使用字串屬性 Name 作為分割索引鍵。在此表格中，儲存您要從中擷取申請人信用評分的所有信用機構的名稱。

1. 登入AWS Management Console並開啟 DynamoDB 支援主控台，網址為 https://console.aws.amazon.com/dynamodb/。
2. 在主控台的導覽窗格中，選擇 [表格]，然後選擇 [建立資料表]。
3. 輸入資料表詳細資訊，如下所示：
   a. 針對資料表名稱，輸入 **GetCreditBureau**。
   b. 針對 Partition key (分割區索引鍵)，輸入 **Name**。
   c. 保留預設選項，然後選擇「建立表格」。
4. 建立表格後，在「表格」清單中選擇GetCreditBureau表格。
5. 選擇「操作」，然后選擇「創建物件」。
6. 在值中，輸入信用機構的名稱。例如：**CredTrack**。
7. 選擇 Create item (建立項目)。
8. 重複此過程並為其他信用機構的名稱創建項目。例如，**KapFinn** 和 **CapTrust**。

步驟 2：更新狀態機器 — 從 DynamoDB 表格擷取結果

在「Step Functions」主控台中，您將新增Task狀態，並使用 AWSSDK 整合，從您在步驟 1 中建立的 DynamoDB 表格擷取信用機構的名稱。您將使用此步驟的輸出作為稍後在本教學課程中新增到工作流程中的Map狀態的輸入。

1. 打開狀CreditCardWorkflow態機進行更新。
2. 選擇 [取得信用機構狀態] 清單。

3. 對於 API 參數，請將表名稱值指定為**GetCreditBureau**。

步驟 3：建立 Lambda 函數，以傳回所有信用機構的信用評分

在此步驟中，您會建立 Lambda 函數，以接收所有信用機構的名稱作為輸入，並傳回每個信用機構的申請人的信用評分。此 Lambda 函數會從您將在本教學課程步驟 4 中新增至工作流程的Map狀態叫用。

1. 創建一個 Node.js 16.x Lambda 函數並將其**get-credit-score**命名。
2. 在標題為的頁面上 get-credit-score，將以下代碼粘貼到「代碼源代碼」區域中。

```
function getScore(arr) {
  let temp;
  let i = Math.floor((Math.random() * arr.length));
  temp = arr[i];
  console.log(i);
  console.log(temp);
  return temp;
}

const arrScores = [700, 820, 640, 460, 726, 850, 694, 721, 556];

exports.handler = (event, context, callback) => {
  let creditScore = getScore(arrScores);
  callback(null, "Credit score pulled is: " + creditScore + ".");
};
```

3. 部署 Lambda 函數。

第 4 步：更新狀態機-添加地圖狀態以迭代獲取信用評分

在「Step Functions 數」主控台中，您可以新增一個Map狀態，以呼叫 get-credit-scoreLambda 函數，以檢查申請人的信用評分，以檢查由「取得信用機構」狀態傳回的所有信用機構的信用評分。

1. 打開狀CreditCardWorkflow態機進行更新。
2. 選擇取得來自所有信用機構州的分數。
3. 在 [組態] 索引標籤中，選擇 [提供項目陣列的路徑]，然後輸入**$.Items**。
4. 選擇「獲取所有分數」進入Map狀態。
5. 在「組態」索引標籤中，確定已選取「整合類型」「最佳化」。

6. 對於函數名稱，請開始輸入 get-credit-scoreLambda 函數的名稱，然後從出現的下拉式清單中選擇它。
7. 對於「承載」，選擇「無承載」。

教學課程 6：儲存工作流程並執行狀態機

現在，您已經在工作流程原型中 AWS 服務 設定了所有使用的資源，您可以將其儲存為 Step Functions 狀態機器，然後開始執行它。

主題

步驟 1：檢閱自動產生的狀態機定義並儲存狀態機

當您將狀態從「流程」索引標籤拖放到工作流程 Studio 中的畫布上以建立工作流程原型時，「Step Functions」會自動即時撰寫工作流程的 Amazon States Language (ASL) 定義。您可以視需要在中編輯此定義程式碼編輯器。

檢閱 ASL 定義並儲存狀態機

1. (選擇性) 選擇上的「定義」Inspector 以檢視狀態機 Amazon States Language (ASL) 定義，該定義會根據您在「動作」和「流程」標籤以及「Inspector 查程式」面板中的選擇自動產生。

   > **Tip**
   >
   > 若要編輯定義，您可以選擇頁面頂端的「程式碼」來開啟程式碼編輯器。在此自學課程中，請繼續使用自動產生的定義。

2. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **`CreditCardWorkflow`**。
3. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器設定中的所有預設選項。

> **Note**
>
> (選擇性) Step Functions 數會自動為狀態機器建立執行角色，具有叫用 `RandomNumberforCredit` Lambda 函數並發佈至 Amazon SNS 主題所需的最低權限。

如果您先前已使用狀態機器的正確許可建立 IAM 角色，並且想要使用它，請在 [權限] 中選取 [選擇現有角色]，然後從清單中選取角色。或選取 [輸入角色 ARN]，然後為該 IAM 角色提供 ARN。

4. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 2：新增剩餘的 IAM 政策

由於 Step Functions 不會自動產生叫用`Parallel`狀態中使用的 Lambda 函數的許可，因此您需要新增必要的原則。

若要新增剩餘的策略

1. 在CreditCardWorkflow頁面上，選擇狀態機器的 IAM 角色以導覽至 IAM 主控台。您將為此頁面上剩餘的 Lambda 函數新增必要的權限。
2. 選擇新增許可，然後選擇連接政策。
3. 在搜尋方塊中，輸入，**`AWSLambdaRole`**然後按 Enter 鍵。
4. 選擇 AWSLambdaRole，然後選擇 [附加原則]。此原則現在已新增至狀態機器的執行角色。此原則可讓您叫用狀態機器中的任何 Lambda 函數。

步驟 3：運行狀態機

狀態機器執行是執行工作流程以執行工作的執行個體。

執行狀態機

1. 在CreditCardWorkflow頁面上，選擇 [開始執行]。

   此時會顯示「開始執行」對話方塊。

2. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

      > **Note**
      >
      > 您不需要提供任何輸入即可執行此狀態機器。但是，如果需要，您可以在其他狀態機器的 [開始執行] 對話方塊的 [輸入] 區域中指定執行輸入。如需如何為狀態機器提供執行輸入的範例，請參閱步驟 4：開始新的執行學習以使用 AWS Step Functions Workflow Studio 教學課程。

   b. 選擇 Start execution (開始執行)。

3. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

教學課程 7：設定輸入和輸出

步驟函數執行會接收 JSON 文字做為輸入，並將該輸入傳遞至工作流程中的第一個狀態。工作流程中的個別狀態會接收 JSON 資料做為輸入，通常會將 JSON 資料作為輸出傳遞至下一個狀態。根據預設，除非您已為工作流程中的一或多個狀態設定輸入和/或輸出，否則資料會從工作流程中的一個狀態

傳遞到下一個狀態。瞭解資訊如何從狀態流向另一個狀態，以及學習如何篩選和操作這些資料，是在 Step Functions 中有效設計和實作工作流程的關鍵。

Step Functions 提供了多個過濾器來控制狀態之間的輸入和輸出數據流。下列篩選器可用於您的工作流程：

> **Note**
>
> 根據您的使用案例，您可能不需要在工作流程中套用所有這些篩選器。

InputPath

選取整個輸入有效負載的 WHERE 部分，以用作工作的輸入。如果您指定此欄位，「步驟函數」會先套用此欄位。

##

指定調用任務之前輸入應該如何看起來像。透過此Parameters欄位，您可以建立索引鍵值配對的集合，這些索引鍵值配對會當做輸入傳遞至AWS 服務整合，例如AWS Lambda函數。這些值可以是靜態的，也可以是從狀態輸入或工作流程前後關聯物件動態選取的。

ResultSelector

決定要從工作輸出中選擇的項目。使用該ResultSelector字段，您可以創建鍵值對的集合，以替換狀態的結果並將ResultPath該集合傳遞給。

ResultPath

決定要放置工作輸出的位置。使用ResultPath來判斷狀態的輸出是其輸入的副本、產生的結果，還是兩者的組合。

OutputPath

決定要傳送至下一個狀態的內容。使用OutputPath，您可以過濾掉不需要的信息，並僅傳遞您關心的 JSON 數據部分。

> **Tip**
>
> Parameters和ResultSelector篩選器的運作方式是建構 JSON，而InputPath和篩選OutputPath器則是透過篩選 JSON 資料物件中的特定節點來運作，而ResultPath篩選器的運作方式是建立可以新增輸出的欄位。

在本教學課程中，您將學習如何執行下列工作：

- 使用 InputPath 濾鏡選擇原始輸入的特定部分
- 使用「參數」篩選器操作選取的輸入
- 使用ResultSelector、ResultPath和OutputPath篩選器設定輸出

若要取得有關在工作流程中配置輸入和輸出的更多資訊，請參閱Step Functions 中的輸入和輸出處理。

使用 InputPath 濾鏡選擇原始輸入的特定部分

使用`InputPath`篩選器選取輸入有效負載的特定部分。

如果未指定`InputPath`，則其值預設為`$`，這會導致狀態的工作參照整個原始輸入，而不是特定部分。

若要瞭解如何使用InputPath篩選器，請執行下列步驟：

- 步驟 1：建立狀態機
- 步驟 2：運行狀態機
- 步驟 3：使用InputPath篩選器選取執行輸入的特定部分

步驟 1：建立狀態機

> **Important**
>
> 確保狀態機與先前建立的 Lambda 函數位於相同的 AWS 帳戶和區域。

1. 使用您在教學課程 4 中學到的`Parallel`狀態範例來建立新的狀態機。請確定您的工作流程原型看起來與下列原型相似。
2. 設定`check-identity`和 `check-address` Lambda 函數的整合。如需建立 Lambda 函數並在狀態機器中使用它們的相關資訊，請參閱步驟 1：建立 Lambda 函數以執行所需的檢查和步驟 2：更新工作流程 — 新增要執行的平行工作。
3. 對於裝載，請確定您保留 [使用狀態輸入作為裝載] 的預設選取項目。
4. 選擇 [下一步]，然後執行教學課程 5 中步驟 1：保存狀態機的步驟 1 到 3，以建立新的狀態機器。在本教學課程中，請為您的狀態機命名**`WorkflowInputOutput`**。

步驟 2：運行狀態機

1. 在WorkflowInputOutput頁面上，選擇 [開始執行]。
2. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

3. 在「輸入」區域中，新增下列 JSON 資料做為執行輸入。

```
{
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    }
  }
}
```

4. 選擇 Start execution (開始執行)。
5. 狀態機器執行會導致錯誤，因為您尚未指定`check-identity`和 `check-address` Lambda 函數必須使用哪些執行輸入部分來執行所需的身分識別和位址驗證。
6. 請繼續執行本自學課程的步驟 3 以修正錯誤。

步驟 3：使用**`InputPath`**篩選器選取執行輸入的特定部分

1. 在 [執行詳細資訊] 頁面上，選擇編輯狀態機器。

2. 若要驗證中提供的執行輸入中提到的申請人身份步驟 2：運行狀態機，請編輯「驗證身份」任務定義，如下所示：

```
...
{
    "StartAt": "Verify identity",
    "States": {
      "Verify identity": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "InputPath": "$.data.identity",
        "Parameters": {
          "Payload.$": "$",
          "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:check-
identity:$LATEST"
        },
        "End": true
      }
    }
  }
  ...
```

因此，下列 JSON 資料會變成可用作check-identity函數的輸入。

```
{
   "email": "jdoe@example.com",
   "ssn": "123-45-6789"
 }
```

3. 若要驗證執行輸入中所述的應徵者地址，請依照下列方式編輯Verify address作業定義：

```
...
{
    "StartAt": "Verify address",
    "States": {
      "Verify address": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "InputPath": "$.data.address",
        "Parameters": {
          "Payload.$": "$",
```

```
          "FunctionName": "arn:aws:lambda:us-east-1:123456789012:function:check-
address:$LATEST"
        },
        "End": true
      }
    }
  }
  ...
```

因此，下列 JSON 資料會變成可用作check-address函數的輸入。

```
{
    "street": "123 Main St",
    "city": "Columbus",
    "state": "OH",
    "zip": "43219"
}
```

4. 選擇 Start execution (開始執行)。狀態機執行現在已成功完成。

使用「參數」篩選器操作選取的輸入

雖然InputPath篩選器可協助您限制您提供的原始 JSON 輸入，但使用Parameters篩選器，您可以傳遞索引鍵值組的集合作為輸入。這些鍵值配對可以是您在狀態機定義中定義的靜態值，也可以是使用InputPath從原始輸入中選取的值。

在您的工作流程中，Parameters會在之後套用InputPath。 Parameters協助您指定基礎工作如何接受其輸入裝載。例如，如果 check-address Lambda 函數接受字串參數做為輸入，而不是 JSON 資料，您可以使用Parameters篩選器來轉換輸入。

在下列範例中，Parameters篩選器會接收您使用 InputPath in 選取的輸入，步驟 3：使用InputPath篩選器選取執行輸入的特定部分並在輸入項目States.Format上套用內建函數，以建立名為的字串。addressString內建函式可協助您在指定輸入上執行基本的資料處理作業。如需詳細資訊，請參閱內部函數。

```
"Parameters": {
    "addressString.$": "States.Format('{}. {}, {} - {}', $.street, $.city, $.state,
 $.zip)"
  }
```

因此，會建立下列字串，並提供給 check-address Lambda 函數做為輸入。

```
{
    "addressString": "123 Main St. Columbus, OH - 43219"
}
```

使用ResultSelector、ResultPath和OutputPath篩選器設定輸出

在WorkflowInputOutput狀態機器中叫用 check-address Lambda 函數時，函數會在執行位址驗證後傳回輸出裝載。在 [執行詳細資料] 頁面上，選擇 [驗證位址] 步驟，並在步驟詳情窗格的 [工作結果] 內檢視輸出承載。

```
{
    "ExecutedVersion": "$LATEST",
    "Payload": {
      "statusCode": 200,
      "body": "{\"approved\":true,\"message\":\"identity validation passed\"}"
    },
    "SdkHttpMetadata": {
      "AllHttpHeaders": {
        "X-Amz-Executed-Version": [
          "$LATEST"
        ],
        ...
        ...
    "StatusCode": 200
  }
```

使用 ResultSelector

現在，如果您需要將身分識別和位址驗證檢查的結果提供給工作流程中的下列狀態，您可以在輸出 JSON 中選取 payload.Body 節點，並使用ResultSelector篩選器中的StringToJson內建函數根據需要將資料格式化。

ResultSelector從工作輸出中選取需要的項目。在下面的例子中，**ResultSelector**採用 $.payload.body 的字符串，並應用**States.StringToJson**內部函數將字符串轉換為 JSON，並將生成的 JSON 放在標識節點內。

```
"ResultSelector": {
    "identity.$": "States.StringToJson($.Payload.body)"
```

```
  }
```

因此，會建立下列 JSON 資料。

```
{
    "identity": {
      "approved": true,
      "message": "Identity validation passed"
    }
  }
```

當您使用這些輸入和輸出篩選器時，您也可能會遇到因為指定了無效的 JSON 路徑運算式而產生的執行階段錯誤。如需詳細資訊，請參閱 。

使用 ResultPath

您可以在初始輸入有效負載中指定位置，以使用ResultPath欄位儲存狀態的任務處理結果。如果未指定ResultPath，則其值預設為$，這會使初始輸入有效負載取代為原始工作結果。如果指定ResultPath為null，則會捨棄原始結果，且初始輸入有效負載會成為有效輸出。

如果您在使用ResultPath欄位建立的 JSON 資料上套用ResultSelector欄位，則工作結果會新增至輸入裝載的結果節點內，如下列範例所示：

```
{
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    },
  "results": {
    "identity": {
      "approved": true
    }
  }
```

```
}
```

使用 OutputPath

您可以在應用程式之後選取狀態輸出的一部分，以傳遞ResultPath至下一個狀態。這可讓您篩選掉不需要的資訊，只傳遞您關注的 JSON 部分。

在下列範例中，OutputPath欄位會將狀態輸出儲存在結果節點內："OutputPath": "$.results"。因此，狀態的最終輸出，您可以傳遞到下一個狀態如下：

```
{
    "addressResult": {
      "approved": true,
      "message": "address validation passed"
    },
    "identityResult": {
      "approved": true,
      "message": "identity validation passed"
    }
}
```

使用控制台功能可視化輸入和輸出數據流

您可以使用 Step Functions 主控台的資料流程模擬器或 [執行詳細資訊] 頁面中的 [進階檢視] 選項，以視覺化方式呈現工作流程中各狀態之間的輸入和輸出資料流

教學課程 8：在主控台中偵錯錯誤

當您使用步驟函數時，您可能會遇到由於原因而產生的執行階段錯誤，例如：

- Choice狀態中「變數」欄位的 JSON 路徑無效。
- 狀態機器定義問題，例如沒有為Choice狀態定義相符的規則。
- 應用過濾器操作輸入和輸出時，JSON 路徑表達式無效。
- 因為 Lambda 函數例外狀況而導致工作失敗。
- IAM 權限錯誤。

在本教學課程中，您將學習如何使用 Step Functions 主控台來偵錯這些錯誤。如需詳細資訊，請參閱Step Functions 中的錯誤處理。

主題

偵錯無效路徑選擇狀態錯誤

當您在狀態的「變數」欄位中指定不正確或無法解析的 JSON 路徑，或者未在Choice狀態中定義相符規則時，您會在Choice執行工作流程時收到錯誤訊息。

為了說明無效路徑錯誤，本教學課程會在您的工作流程中引入Choice狀態錯誤。您將使用狀CreditCardWorkflow態機並編輯其定義以引入錯誤。

1. 開啟 Step Functions 主控台，然後選擇狀CreditCardWorkflow態機器。
2. 選擇編輯以編輯狀態機定義。使更改在下面的代碼中突出顯示您的狀態機定義。

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Get credit limit",
  "States": {
    "Get credit limit": {
      ...
      ...
    },
    "Credit applied >= 5000?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.Payload",
          "NumericLessThan": 5000,
          "Next": "Auto-approve limit"
        },
        {
          "Variable": "$.Payload",
          "NumericGreaterThanEquals": 5000,
          "Next": "Wait for human approval"
        }
        ],
        "Default": "Wait for human approval"
    },
    ...
    ...
```

```
  }
}
```

3. 選擇「儲存」，然後選擇「儲存」。
4. 運行狀態機。
5. 在狀態機器執行的 [執行詳細資訊] 頁面上，執行下列其中一個動作：

   a. 在錯誤訊息上選擇「原因」，以檢視執行失敗的原因。

   b. 選擇在錯誤訊息上顯示步驟詳細資訊，以檢視造成錯誤的步驟。
6. 在 [步驟詳細資訊] 區段的 [輸入和輸出] 索引標籤中，選擇 [進階] 檢視切換按鈕，以查看所選狀態的輸入和輸出資料傳輸路徑。
7. 在「圖表」檢視中，確定已套用「點數」大於等於 5000？ 已選取並執行下列動作：

   a. 在輸入框中查看狀態的輸入值。

   b. 選擇 [定義] 索引標籤，並注意為 [變數] 欄位指定的 JSON 路徑。

      申請信貸的輸入值大於等於 5000？ state 是一個數值，而你已經將輸入值的 JSON 路徑指定為`$.Payload`。在狀態機執行期間，狀`Choice`態無法解析此 JSON 路徑，因為它不存在。
8. 編輯狀態機以將「變數」欄位值指定為**$**。

```
{
  "Comment": "A description of my state machine",
  "StartAt": "Get credit limit",
  "States": {
    "Get credit limit": {
      ...
      ...
    },
    "Credit applied >= 5000?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$",
          "NumericLessThan": 5000,
          "Next": "Auto-approve limit"
        },
        {
          "Variable": "$",
          "NumericGreaterThanEquals": 5000,
          "Next": "Wait for human approval"
```

```
        }
        ],
        "Default": "Wait for human approval"
    },
    ...
    ...
  }
}
```

在應用輸入和輸出過濾器時調試 JSON 路徑表達式錯誤

當您使用輸入和輸出篩選器時，您可能會遇到因為指定了無效的 JSON 路徑運算式而產生的執行階段錯誤。

下列範例使用您在教學課程 5 中建立的WorkflowInputOutput狀態機器，並示範您使用`ResultSelector`篩選器來選取工作輸出部分的案例。

1. 套用`ResultSelector`篩選器，為 [驗證身分] 步驟選擇工作輸出的一部分。若要這麼做，請依照下列方式編輯您的狀態機定義：

```
{
  "StartAt": "Verify identity",
  "States": {
    "Verify identity": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:check-
identity",
        "Payload": {
          "email": "jdoe@example.com",
          "ssn": "123-45-6789"
        }
      },
      ...
      ...
      "ResultSelector": {
        "identity.$": "$.Payload.body.message"
 }",
      "End": true
    }
  }
```

```
}
```

2. 運行狀態機。
3. 在狀態機器執行的 [執行詳細資訊] 頁面上，執行下列動作：
   a. 在錯誤訊息上選擇「原因」，以檢視執行失敗的原因。
   b. 選擇在錯誤訊息上顯示步驟詳細資訊，以檢視造成錯誤的步驟。
4. 在錯誤訊息中，請注意 $.Payload.body 節點的內容是逸出的 JSON 字串。發生錯誤是因為您無法使用 JSON 路徑標記法參考字串。
5. 若要參照 $.Payload.body. 訊息節點，請執行下列動作：
   a. 使用States.StringToJSON內部函數首先將字符串轉換為 JSON 格式。
   b. 在內建函數內指定 JSON 路徑。

```
"ResultSelector": {
 "identity.$":"States.StringToJson($.Payload.body.message)"
}
```

6. 再次運行狀態機。

使用案例

AWS Step Functions可讓您建置視覺化工作流程，協助快速將業務需求轉換為應用程式 Step Functions 會為您管理狀態、檢查點和重新啟動，並提供內建功能以自動處理錯誤和例外狀況。為了更好地了解 Step 函數可以為您提供的功能，請閱讀以下使用案例：

主題

- 資料處理
- 機器學習
- 微服務協調
- IT 和安全自動化

資料處理

隨著資料量的增加，來自日益多樣化的來源，組織發現他們需要快速移動來處理這些資料，以確保他們做出更快、明智的業務決策。若要大規模處理資料，組織需要彈性佈建資源，以管理從行動裝置、應用程式、衛星、行銷和銷售、營運資料存放區、基礎架構等接收到的資訊。

Step Functions 提供成功管理資料處理工作流程所需的可擴充性、可靠性和可用性。您可以使用 Step Functions 來管理數百萬個並行執行，因為它可以水平擴展並提供容錯工作流程。使用平行執行 (例如 Step Functions 的狀態類型)，或使用其##Map狀態類型的動態平行處理作業，更快地處理資料。作為工作流程的一部分，您可以使用Map狀態對靜態資料存放區 (例如 Amazon S3 儲存貯體) 中的物件進行迭代。Step Functions 也可讓您輕鬆重試失敗的執行，或選擇特定的方式來處理錯誤，而不需要管理複雜的程序。

根據您的資料處理需求，Step Functions 會直接與其他資料處理服務整合，AWS例如用AWS Batch於批次處理、用於大數據處理的 Amazon EMR、用AWS Glue於資料準備的 Athena 進行資料分析，以及用AWS Lambda於運算。

客戶使用 Step Functions 完成的資料處理工作流程類型範例包括：

文件，視頻和圖像處理

- 拍攝一系列視頻文件並將其轉換為其他尺寸或分辨率，這些文件非常適合顯示它們的設備，例如手機，筆記本電腦或電視。
- 拍攝用戶上傳的大量照片，並將其轉換為縮略圖或各種分辨率圖像，然後可以在用戶的網站上顯示。

- 擷取半結構化資料 (例如 CSV 檔案)，並將其與非結構化資料 (例如發票) 結合使用，以產生每月傳送給業務利害關係人的業務報告。
- 從衛星收集到的地球觀測數據，將其轉換為彼此一致的格式，然後添加在地球上收集的其他數據源以獲得更多見解。
- 從各種運輸模式中獲取產品的運輸日誌，並使用蒙特卡羅模擬查找優化，然後將報告發送回依靠您運送貨物的組織和人員。

坐標提取，轉換和加載（ETL）工作：

- 透過使用的一系列資料準備步驟，將銷售機會記錄與行銷量度資料集結合在一起AWS Glue，並產生可在整個組織中使用的商業智慧報告。
- 建立、啟動和終止用於大數據處理的 Amazon EMR 叢集。

批次處理和高效能運算 (HPC) 工作負載：

- 構建基因組學二次分析管道，將原始的整個基因組序列處理為變體調用。將原始文件與參考序列對齊，並使用動態並行性調用指定染色體列表上的變體。
- 通過使用不同的電氣和化學化合物模擬各種佈局，找到生產下一個移動設備或其他電子設備的效率。透過各種模擬執行大量的工作負載批次處理，以取得最佳設計。

機器學習

機器學習可讓組織快速分析收集到的資料以識別模式，然後以最少的人為介入做出決策。機器學習始於一組初始的資料，稱為訓練資料。此訓練資料有助於提高機器學習模型的預測準確度，並作為此模型學習的基礎。一旦模型被認為足夠準確以滿足業務需求，它就會部署到生產環境中。AWS Step Functions資料科學軟體開發套件 (SDK) 是開放原始碼程式庫，可讓您輕鬆建立工作流程，以預先處理資料、訓練模型，然後使用 Amazon SageMaker 和 Step Functions 發佈模型。

預先處理現有資料集是組織經常建立訓練資料的方式。此方法會新增資訊，例如標示影像中的物件、註解文字或處理音訊。若要預先處理您可以使用的資料AWS Glue，或者您可以建立執行 Jupyter SageMaker Notebook 應用程式的筆記本執行個體。資料準備就緒後，即可將其上傳到 Amazon S3 以便輕鬆存取。由於機器學習模型經過訓練，您可以對每個模型的參數進行調整，以提高準確性，直到準備好部署為止。

Step Functions 可讓您在SageMaker上協調端對端機器學習工作流程。這些工作流程可以包括資料預處理、後處理、特徵工程、資料驗證和模型評估。將模型部署到生產環境後，您可以優化和測試新方

法，以持續改善業務成果。您可以直接在 Python 中建立可供生產使用的工作流程，也可以使用 Step Functions 資料科學 SDK 複製該工作流程、試驗新選項，並將精細的工作流程置於生產環境中。

客戶使用 Step Functions 的某些機器學習工作流程類型包括：

詐騙偵測

- 識別並防止詐騙交易發生，例如信用欺詐。
- 使用訓練有素的機器學習模型偵測並防止帳戶遭到盜用。
- 識別促銷濫用行為，包括建立虛假帳戶，以便您快速採取行動。

個人化與建議

- 根據預測吸引他們的興趣，向目標客戶推薦產品。
- 預測客戶是否要將其帳戶從免費方案升級為付費訂閱。

資料豐富

- 使用資料擴充作為預先處理的一部分，為更準確的機器學習模型提供更好的訓練資料。
- 註釋文本和音頻摘錄以添加語法信息，例如諷刺和俚語。
- 在影像中標示其他物件，以提供重要資訊以供模型學習，例如物件是蘋果、籃球、岩石還是動物。

微服務協調

微服務架構將應用程式分解為鬆散耦合的服務。優點包括改善的可擴充性、提高彈性，以及加快產品上市時間。每個微服務都是獨立的，因此無需擴展整個應用程式即可輕鬆擴展單一服務或功能。個別服務是鬆散結合的，讓獨立團隊專注於單一業務流程，而不需要他們瞭解整個應用程式。微型服務也可讓您選擇哪些個別元件符合您的業務需求，讓您彈性變更選擇，而無需重新撰寫整個工作流程。不同的團隊可以使用他們選擇的程式設計語言和架構來處理他們的微服務，而且這個微服務仍然可以透過應用程式設計介面 (API) 與應用程式中的任何其他人通訊。

Step Functions 提供多種管理微服務工作流程的方法。對於長期執行的工作流程，您可以將標準工作流程與AWS Fargate整合搭配使用，協調在容器中執行的應用 對於需要立即回應的短期大量工作流程，同步 Express 工作流程是理想的選擇。這些可用於基於 Web 的或移動應用程序，這些應用程序通常具有短持續時間的工作流程，並且在返回響應之前需要完成一系列步驟。您可以直接從 Amazon API 閘道觸發同步快速工作流程，連線會保持開啟，直到工作流程完成或逾時為止。對於不需要立即回應的短期工作流程，Step Functions 提供非同步 Express 工作流程。

使用步驟函式的某些 API 協調流程範例包括：

同步或即時工作流程

- 變更記錄中的值，例如更新員工的姓氏，並讓變更立即顯示在螢幕上。
- 在結帳期間更新訂單，例如新增、移除或變更料號數量，然後立即將更新反映回客戶。
- 執行快速處理工作，並立即將結果傳回給要求者。

容器協調

- 使用 Amazon 彈性 Kubernetes 服務或亞馬遜彈性容器服務 (ECS) 在 Kubernetes 上執行任務，並與其他服AWS務整合，例如透過 Amazon SNS 傳送通知，做為相同工作流程的一部分。

IT 和安全自動化

IT 自動化可協助管理日益複雜且耗時的作業，例如升級和修補軟體、部署安全性更新以解決弱點、選取基礎結構、同步處理資料、路由支援票證等等。重複且耗時的工作自動化可讓您的組織大規模快速且一致地完成例行作業。這可讓您專注於策略性工作，例如功能開發、複雜的支援要求和創新，同時滿足這些不斷成長的需求。

Step Functions 可讓您建立可自動擴展以符合業務需求的工作流程，而無需手動介入。在工作流程中發生錯誤的情況下，通常不需要手動介入。Step Functions 可讓您自動重試失敗的工作，以及可管理工作流程錯誤的指數輪詢。

在某些情況下，需要人工介入才能進行工作流程。例如，批准大幅增加信貸可能需要人工批准。若要管理此項目，您可以在 Step Functions 中定義分支邏輯，以便只有超過定義數量的請求才需要人工核准，而所有其他請求則會自動完成。在需要人工核准的情況下，Step Functions 可讓您在特定步驟暫停工作流程、等待回應，然後在收到回應後繼續工作流程。

客戶使用 Step Functions 的自動化工作流程類型的一些範例包括：

IT 自動化

- 自動修復事件，例如開啟 SSH 連接埠、磁碟空間不足，或是將公開存取權授予 Amazon S3 儲存貯體時。
- 自動化的部署 AWS CloudFormation StackSets

安全自動化

- 自動回應已公開使用者和使用者存取金鑰的案例。
- 根據定義的原則動作 (例如限制特定 ARN 的動作或套用其他動作)，自動修復安全性事件回應。
- 在收到網絡釣魚電子郵件後幾秒鐘內警告員工。

人工批准

- 將機器學習模型的訓練自動化，然後需要資料科學家手動核准模型，然後再根據收到的回應自動部署或拒絕模型。
- 根據情緒分析自動化收到的客戶意見反應路由，以便具有負面情緒的人可立即升級以進行手動審核。

Step Functions 的工作原理

本節說明的重要概念可幫助您熟悉 AWS Step Functions 並了解其運作方式。

主題

標準與快速工作流程

建立狀態機時，您可以選取「標準」或「快速」的「類型」。狀態機器的預設「類型」為「標準」。「類型」為「標準」的狀態機稱為「標準」工作流程，而「類型」為「快速」的狀態機稱為 Express 工作流程。

對於標準工作流程和 Express 工作流程，您可以使用定義狀態機器Amazon States Language。根據您選取的類型，狀態機器執行的行為會有所不同。

> **Important**
>
> 建立狀態機後，您選擇的「類型」無法變更。

> **Note**
>
> 如果您在 Step Functions 的控制台之外定義狀態機器，例如在您選擇的編輯器中，則必須使用擴展名為. asl.json 來保存狀態機定義。

標準工作流程非常適合長時間執行 (長達一年)、耐用且可稽核的工作流程。您可以使用 Step Functions API 擷取完整的執行歷史記錄，最多可在執行完成後 90 天。標準工作流程只遵循一次的模型，除非您在 ASL 中指定了行為，否則您的任務和狀態永遠不會執`Retry`行一次以上。這使得標準工作流程適合協調非冪等動作，例如啟動 Amazon EMR 叢集或處理付款。標準工作流程執行會根據處理的狀態轉換數量計費。

快速工作流程適合大量事件處理工作負載，例如 IoT 資料擷取、串流資料處理和轉換，以及行動應用程式後端。這種工作流程最多可以執行五分鐘。Express 工作流程採用at-least-once模型，其中執行可能會執行多次。這使得 Express 工作流程非常適合協調冪等動作，例如在 Amazon DynamoDB 中透過 PUT 動作轉換輸入資料和存放。Express Workflow 執行會依執行次數、執行持續時間，以及執行執行時耗用的記憶體計費。

標準和快速工作流程可以自動啟動，以回應事件，例如來自 Amazon API Gateway 的 HTTP 請求 (大規模完全受管 API)、IoT 規則以及 Amazon EventBridge 中 140 多個其他事件來源。

> **Tip**
>
> 若要將快速工作流程的範例部署到您的AWS 帳戶，請參閱模組 7-API Gateway、平行狀態、AWS Step Functions工作坊的快速工作流程。

如需「標準」和「快速工作流程」執行之主控台體驗的相關資訊，請參閱主控台中的標準和快速工作流程執行。

標準與快速工作流程

	標準工作流程	快速工作流程：同步和非同步
持續時間上限	一年	五分鐘
支援的執行啟動速率	如需與支援的執行開始率相關配額的詳細資訊，請參閱與 API 動作節流相關的配額。	如需與支援的執行開始率相關配額的詳細資訊，請參閱與 API 動作節流相關的配額。

	標準工作流程	快速工作流程：同步和非同步
支援的狀態轉換速率	如需與支援的狀態轉換率相關配額的資訊，請參閱與狀態節流有關的配額。	沒有限制
定價	按狀態轉換的數量定價。每次執行中的步驟完成時，就會計算狀態轉換。	依您執行的執行次數、其持續時間和記憶體用量來計費。
執行歷史記錄	可以使用 Step Functions API 列出和描述執行。可以通過控制台直觀地調試執行。您也可以在狀態機器上啟用記錄，在記錄 CloudWatch 檔中檢查這些資料。 如需有關在主控台中偵錯標準工作流程執行的詳細資訊，請參閱主控台中的標準和快速工作流程執行和檢視和偵錯執行。	無限制的執行歷史記錄，也就是說，您可以在 5 分鐘內產生的多個執行歷史記錄項目。 透過在狀態機器上啟用記錄，可以在記錄 CloudWatch 檔或 Step Functions 主控台中檢查執行。 如需有關在主控台中偵錯 Express 工作流程執行的詳細資訊，請參閱主控台中的標準和快速工作流程執行和檢視和偵錯執行。
執行語義	只有一次工作流程執行。	非同步快速工作流程：t-least-once工作流程執行。 同步快速工作流程：t-most-once工作流程執行。

	標準工作流程	快速工作流程：同步和非同步
服務整合	支援所有服務整合與模式。	支援所有服務整合。 **Note** Express 工作流程不支援工作執行 (`.sync`) 或回呼 (`.waitForTaskToken`) 服務整合模式。
Step Functions 活動	支援 Step Functions 活動。	不支援 Step Functions 活動。

同步和非同步快速工作流

您可以選擇兩種類型的 Express 工作流程：非同步快速工作流程和同步 Express 工作流程。

- 非同步 Express 工作流程會傳回工作流程已啟動的確認，但不要等待工作流程完成。若要取得結果，您必須輪詢服務的CloudWatch 記錄檔。當您不需要立即回應輸出 (例如訊息服務或其他服務不依賴的資料處理) 時，您可以使用非同步 Express 工作流程。您可以啟動非同步 Express 工作流程來回應事件、Step Functions 中的巢狀工作流程，或使用 `StartExecution` API 呼叫。
- 同步 Express 工作流程會啟動工作流程，等到它完成，然後傳回結果。同步 Express 工作流程可用於協調微服務。使用同步 Express 工作流程，您可以開發應用程式，而不需要開發額外的程式碼來處理錯誤、重試或執行 parallel 工作。您可以執行從 Amazon API Gateway 叫用的同步快速工作流程AWS Lambda，或使用 `StartSyncExecution` API 呼叫。

> **Note**
>
> 如果您從主控台同步執行 Step Functions Express 工作流程，則`StartSyncExecution`要求會在 60 秒後經過。若要同步執行 Express 工作流程，最長可達五分鐘，`StartSyncExecution`請使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 而非 Step Functions 式主控台提出要求。

同步快速執行 API 呼叫不會造成現有帳戶容量限制。Step Functions 可依需求提供容量，並隨著持續的工作負載自動擴充 工作負載激增可能會受到限制，直到容量可用為止。

執行保證

標準工作流程	異步快速工作流	同步快速工作流	
只執行一次工作流程	t-least-once工作流程執行	t-most-once工作流程執行	
狀態轉換之間在內部持續執行狀態。	狀態轉換之間的執行狀態不會持續存在。	狀態轉換之間的執行狀態不會持續存在。	
在啟動與目前正在執行的工作流程相同名稱的執行時，自動傳回冪等回應。新的工作流程不會啟動，一旦目前正在執行的工作流程完成，就會擲回例外狀況。	冪等不會自動管理。啟動多個具有相同名稱的工作流程會導致並行執行。如果狀態機器邏輯不是冪等的，可能會導致內部工作流程狀態遺失。	冪等不會自動管理。Step Functions 等待一旦執行開始，並在完成後返回狀態機的結果。如果發生例外，工作流程不會重新啟動。	
執行歷程記錄資料會在 90 天後移除。移除 out-of-date 執行資料後，可重複使用工作流程名稱。 若要符合法規遵循、組織或法規需求，您可以傳送配額要求，將執行歷程記錄保留期限縮短為 30 天。若要這麼做，請使用 AWS Support Center	Step Functions 不會擷取執行歷史記錄。必須透過 Amazon CloudWatch 日誌啟用記錄功能。	Step Functions 不會擷取執行歷史記錄。必須透過 Amazon CloudWatch 日誌啟用記錄功能。	

標準工作流程	異步快速工作流	同步快速工作流	
Console並建立新案例。			

使用快速工作流程最佳化

Step Functions 會根據您用來建置狀態機器的工作流程類型來決定標準和快速工作流程的定價。若要最佳化無伺服器工作流程的成本，您可以遵循下列其中一項或兩項建議：

主題

有關選擇「標準」或「快速」工作流程類型如何影響帳單的資訊，請參閱AWS Step Functions定價。

秘訣 #1: 在標準工作流程內嵌快速工作流程

Step Functions 會執行具有有限持續時間和步驟數目的工作流程。某些工作流程可能會在短時間內完成執行。其他人可能需要長時間執行和 high-event-rate 工作流程的組合。使用 Step Functions，您可以從多個較小、更簡單的工作流程中建立大型、複雜的工作流程。

例如，若要建立訂單處理工作流程，您可以將所有非冪等作業納入「標準」工作流程中。這可能包括操作，例如通過人工互動批准訂單和處理付款。然後，您可以在 Express 工作流程中結合一系列冪等動作，例如傳送付款通知和更新產品庫存。您可以在「標準」工作流程中嵌套此 Express 工作流程。在此範例中，「標準」工作流程稱為父狀態機器。巢狀 Express 工作流程稱為子狀態機器。

秘訣 #2: 將標準工作流程轉換為 Express 工作流程

如果現有的「標準」工作流程符合下列需求，您可以將它們轉換為 Express 工作流程：

- 工作流程必須在五分鐘內完成其執行。
- 工作流程符合at-least-once執行模型。這表示工作流程中的每個步驟可能會執行一次以上。
- 工作流程不使用`.waitForTaskToken`或`.sync`服務整合模式。

> **Important**
>
> 快速工作流程使用 Amazon CloudWatch 日誌記錄執行歷史記錄。使用 CloudWatch 記錄時會產生額外費用。

使用主控台將標準工作流程轉換為 Express 工作流程

1. 開啟「Step Functions」主控台。
2. 在 [狀態電腦] 頁面上，選擇 [標準類型狀態機器] 以開啟它。

   > **Tip**
   >
   > 從 [任何類型] 下拉式清單中，選擇 [標準] 篩選狀態機器清單，並僅檢視 [標準] 工作流程。

3. 選擇「複製到新的」。

   工作流 Studio 會在設計模式顯示您選取的狀態機的工作流程中開啟。
4. (選擇性) 更新工作流程設計。
5. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。
6. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   請確定針對「類型」選擇「快速」。保留狀態機器設定上的所有其他預設選項。

   > **Note**
   >
   > 如果您要轉換先前在AWS CDK或中定義的標準工作流程AWS SAM，則必須變更的值`Type`和 `Resource` name。

7. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

如需管理工作流程成本最佳化時的最佳做法和指導方針的詳細資訊，請參閱建立具成本效益的AWS Step Functions工作流程

狀態

個別狀態可以根據其輸入做出決策、從這些輸入執行動作，以及將輸出傳遞給其他狀態。在中AWS Step Functions，您可以使用亞馬遜州語言 (ASL) 定義工作流程。Step Functions 主控台提供狀態機器的圖形表示，以協助視覺化應用程式的邏輯。

> **Note**
>
> 如果您在 Step Functions 的控制台之外定義狀態機器，例如在您選擇的編輯器中，則必須使用擴展名為. asl.json 來保存狀態機定義。

狀態是您的狀態機器中的元素。狀態會使用自身的名稱來稱呼，這可以是任何字串，但在整個狀態機器範圍內必須是唯一的。

狀態可以在您的狀態機器中執行各種功能：

- 在您的狀態機器上執行一些作業 (Task 狀態)。
- 在要執行的分支之間進行選擇 (Choice 狀態)
- 故障或成功時停止執行 (Fail 或 Succeed 狀態)
- 將其輸入傳遞至其輸出，或將某些固定資料插入工作流程 (「通過」狀態)
- 提供一定時間的延遲，或直到指定的日期和時間 (「等待」狀態)
- 開始平行分支執行 (Parallel 狀態)
- 動態反覆運算步驟 (Map 狀態)

以下是名為 `HelloWorld` 的範例狀態，其會執行 AWS Lambda 函數。

```
"HelloWorld": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
  "Next": "AfterHelloWorldState",
  "Comment": "Run the HelloWorld Lambda function"
}
```

狀態有許多常見的功能：

- 一個Type字段，指示它是什麼類型的狀態。
- 一個可選Comment字段，用於保存有關狀態的人類可讀註釋或描述。
- 每個狀態 (但 Succeed 或 Fail 狀態除外) 都需要有 Next 欄位，或也可以透過指定 End 欄位成為結束狀態。

> **Note**
>
> Choice 狀態可能會擁有多於一個 Next，但是每個「選擇規則」內只會有一個。
> 狀Choice態無法使用End。

某些狀態類型需要額外的欄位，或者可能會重新定義常見欄位的用法。

建立並執行標準工作流程之後，您可以檢視 Step Functions 主控台中的「執行詳細資訊」頁面，存取每個狀態、其輸入與輸出、作用中的時間以及持續時間的相關資訊。如需詳細資訊，請參閱在 Step Functions 主控台上檢視和偵錯執行。

建立並執行快速工作流程執行後，如果已啟用快速工作流程的記錄功能，您可以在 Amazon Lo CloudWatch gs 或 Step Functions 主控台中存取有關執行的資訊。如需詳細資訊，請參閱 在 Step Functions 主控台上檢視和偵錯執行。

主題

- Amazon States Language
- Pass
- 任務
- Choice
- 等候
- Succeed

- Fail
- 平行
- Map

Amazon States Language

Amazon States Language 是一種以 JSON 為基礎的結構化語言，用於定義狀態機器、可執行工作 (狀態) 的Task狀態集合、決定要轉換到下一個 (Choice狀態) 的狀態、停止執行並出現錯誤 (Fail狀態) 等等。

如需詳細資訊，請參閱 Amazon 狀態語言規格和 Statelint，這是一種驗證 Amazon 狀態語言程式碼的工具。

若要使用 Amazon 州語言在 Step 函數主控台上建立狀態機器，請參閱入門。

> **Note**
>
> 如果您在 Step Functions 的控制台之外定義狀態機器，例如在您選擇的編輯器中，則必須使用擴展名為. asl.json 來保存狀態機定義。

示例亞馬遜國家語言規範

```
{
  "Comment": "An example of the Amazon States Language using a choice state.",
  "StartAt": "FirstState",
  "States": {
    "FirstState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type" : "Choice",
      "Choices": [
        {
          "Variable": "$.foo",
          "NumericEquals": 1,
          "Next": "FirstMatchState"
```

```
        },
        {
          "Variable": "$.foo",
          "NumericEquals": 2,
          "Next": "SecondMatchState"
        }
      ],
      "Default": "DefaultState"
    },

    "FirstMatchState": {
      "Type" : "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:OnFirstMatch",
      "Next": "NextState"
    },

    "SecondMatchState": {
      "Type" : "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:OnSecondMatch",
      "Next": "NextState"
    },

    "DefaultState": {
      "Type": "Fail",
      "Error": "DefaultStateError",
      "Cause": "No Matches!"
    },

    "NextState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
      "End": true
    }
  }
}
```

主題

- 国家机结构
- 內部函數
- 常見狀態欄位

国家机结构

狀態機器是使用 JSON 文字來定義的，而該 JSON 文字代表包含下列欄位的結構。

Comment (選用)

狀態機器的人類可讀描述。

StartAt (必要)

一個字串，該字串必須完全符合其中一個狀態物件的名稱 (區分大小寫)。

TimeoutSeconds(選擇性)

狀態機器可以執行的執行秒數上限。如果執行超過指定的時間，執行會失敗，出現States.Timeout 錯誤名稱。

Version (選用)

狀態機器中使用的 Amazon 州語言版本 (默認為「1.0」) 。

States (必要)

一個物件，其中包含以逗號分隔的一組狀態。

該States字段包含狀態。

```
{
    "State1" : {
    },

    "State2" : {
    },
    ...
}
```

狀態機器是由其包含的狀態和其間的關係所定義。

以下是範例。

```
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass state",
  "StartAt": "HelloWorld",
  "States": {
```

```
    "HelloWorld": {
      "Type": "Pass",
      "Result": "Hello World!",
      "End": true
    }
  }
}
```

啟動此狀態執行時，系統會從 StartAt 欄位 ("HelloWorld") 中參考的狀態著手。如果此狀態具有 "End": true 欄位，則執行停止並傳回結果。否則，系統會尋找 "Next": 欄位並繼續以該狀態執行。此程序會重複執行，直到系統達到結束狀態 (具有 "Type": "Succeed"、"Type": "Fail" 或 "End": true 的狀態)，或發生執行時間錯誤為止。

下列規則適用於狀態機器內的狀態：

- 封閉區塊內的狀態可以任何順序發生，但是其列出的順序不會影響其執行順序。此順序是由狀態本身的內容所決定。
- 在狀態機器內，只能有一個狀態指定為 start 狀態，而該狀態是由最上層結構中 StartAt 欄位的值所指定。此狀態是開始執行時最先執行的狀態。
- End 欄位為 true 的任何狀態都會被視為 end (或 terminal) 狀態。根據您的狀態機器邏輯 (例如，如果狀態機具有多個執行分支) ，您可能會有多個狀態。end
- 如果狀態機器只包含一個狀態，該狀態可能同時是 start 狀態和 end 狀態。

內部函數

Amazon States 語言提供數個內建函數 (也稱為內建函式)，可協助您在不使用狀態的情況下執行基本資料處理操作。Task內在函數是看起來類似於編程語言中的函數的構造。它們可用於幫助有效負載構建器處理進出狀Task態Resource字段的數據。

在 Amazon States 語言中，內建函數會根據您要執行的資料處理任務類型，分為下列類別：

- 數組的內在函數
- 數據編碼和解碼的內在函數
- 散列計算的內在
- JSON 數據操作的內在函數
- 數學運算的內在函數
- 字符串操作的內在

- 唯一標識符生成的內在
- 泛型操作的內在

> **Note**
>
> - 若要使用內建函數，您必須.$在狀態機器定義的索引鍵值中指定，如下列範例所示：
>
> ```
> "KeyId.$": "States.Array($.Id)"
> ```
>
> - 您最多可以在工作流程的欄位內嵌 10 個內建函數。下列範例顯示名為的欄位包*myArn*含九個巢狀內建函式：
>
> ```
> "myArn.$": "States.Format('{}.{}.{}',
>  States.ArrayGetItem(States.StringSplit(States.ArrayGetItem(States.StringSplit($.ImageRe
>  '/'), 2), '.'), 0),
>  States.ArrayGetItem(States.StringSplit(States.ArrayGetItem(States.StringSplit($.ImageRe
>  '/'), 2), '.'), 1))"
> ```

> **Tip**
>
> 如果您在本機開發環境中使用 Step Functions，請確定您使用的是 1.12.0 或更高版本，以便能夠在工作流程中包含所有內建函數。

支援內建函數的欄位

下表顯示哪些欄位支援每個狀態的內建函式。

支援內建函數的欄位

State	通行證	任務	選擇	等候	成功	失敗	平行	Map
InputPath								
參數	✓	✓					✓	✓

State

	通行證	任務	選擇	等候	成功	失敗	平行	Map
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
變數								
<Comparison Operator> 路徑								
TimeoutSecondsPath								
HeartbeatSecondsPath								
憑證		✓						

數組的內在函數

使用下列內建函式來執行陣列操作。

`States.Array`

`States.Array`內在函數接受零個或多個參數。解釋器返回一個 JSON 數組，其中包含按照提供的順序的參數的值。例如，假設有下列輸入：

```
{
  "Id": 123456
```

```
}
```

你可以使用

```
"BuildId.$": "States.Array($.Id)"
```

這將返回以下結果：

```
“BuildId”: [123456]
```

States.ArrayPartition

使用States.ArrayPartition內建函式來分割大型陣列。您也可以使用此內建來切片資料，然後以較小的區塊傳送有效負載。

這個內在函數有兩個參數。第一個參數是一個數組，而第二個參數定義塊大小。解釋器將輸入數組塊成由塊大小指定的大小的多個數組。如果陣列中剩餘項目的數目小於區塊大小，則最後一個陣列區塊的長度可能會小於前一個陣列區塊的長度。

輸入驗證

- 您必須指定陣列作為函數第一個引數的輸入值。
- 您必須為代表區塊大小值的第二個引數指定非零的正整數。

  如果您為第二個引數指定非整數值，Step Functions 會將其四捨五入為最接近的整數。
- 輸入陣列不能超過 256 KB 的步進函數的裝載大小限制。

例如，給定以下輸入數組：

```
{"inputArray": [1,2,3,4,5,6,7,8,9] }
```

您可以使用該States.ArrayPartition函數將數組分成四個值的塊：

```
"inputArray.$": "States.ArrayPartition($.inputArray,4)"
```

這將返回以下數組塊：

```
{"inputArray": [ [1,2,3,4], [5,6,7,8], [9]] }
```

在前面的例子中，States.ArrayPartition函數輸出三個數組。前兩個陣列各包含四個值，如區塊大小所定義。第三個數組包含剩餘值，並且小於定義的塊大小。

States.ArrayContains

使用States.ArrayContains內建函式來判斷陣列中是否存在特定值。例如，您可以使用此函數來偵測Map狀態反覆運算中是否有錯誤。

這個內在函數有兩個參數。第一個參數是一個數組，而第二個參數是數組中要搜索的值。

輸入驗證

- 您必須指定一個數組作為函數的第一個參數的輸入值。
- 您必須指定有效的 JSON 物件作為第二個引數。
- 輸入陣列不能超過 256 KB 的步進函數的裝載大小限制。

例如，給定以下輸入數組：

```
{
   "inputArray": [1,2,3,4,5,6,7,8,9],
   "lookingFor": 5
}
```

您可以使用該States.ArrayContains函數來查找以下內容中的lookingFor值inputArray：

```
"contains.$": "States.ArrayContains($.inputArray, $.lookingFor)"
```

由於儲存在中的值包含lookingFor在中inputArray，因此會States.ArrayContains傳回下列結果：

```
{"contains": true }
```

States.ArrayRange

使用States.ArrayRange內建函式建立包含特定範圍元素的新陣列。新陣列最多可包含 1000 個元素。

這個函數有三個參數。第一個參數是新數組的第一個元素，第二個參數是新數組的最後一個元素，第三個參數是新數組中元素之間的增量值。

輸入驗證

- 您必須為所有引數指定整數值。

  如果您為任何引數指定非整數值，Step Functions 會將其四捨五入為最接近的整數。
- 您必須為第三個引數指定非零值。
- 新產生的陣列不能包含超過 1000 個項目。

例如，下面使用該States.ArrayRange函數將創建一個數組，其第一個值為 1，最終值為 9，並且第一個值和最終值之間的值為每個項目增加兩個：

```
"array.$": "States.ArrayRange(1, 9, 2)"
```

這將返回以下數組：

```
{"array": [1,3,5,7,9] }
```

States.ArrayGetItem

這個內在函數返回指定索引的值。這個函數有兩個參數。第一個參數是值的數組，第二個參數是要返回的值的數組索引。

例如，使用下列inputArray和index值：

```
{
   "inputArray": [1,2,3,4,5,6,7,8,9],
   "index": 5
}
```

從這些值中，您可以使用States.ArrayGetItem函數返回數組中index位置 5 的值：

```
"item.$": "States.ArrayGetItem($.inputArray, $.index)"
```

在這個例子中，States.ArrayGetItem將返回以下結果：

```
{ "item": 6 }
```

States.ArrayLength

States.ArrayLength內在函數返回一個數組的長度。它有一個參數，該數組返回的長度。

例如，給定以下輸入數組：

```
{
   "inputArray": [1,2,3,4,5,6,7,8,9]
}
```

您可以使States.ArrayLength用返回的長度inputArray：

```
"length.$": "States.ArrayLength($.inputArray)"
```

在此範例中，States.ArrayLength會傳回下列 JSON 物件，代表陣列長度：

```
{ "length": 9 }
```

States.ArrayUnique

States.ArrayUnique內在函數從數組中刪除重複的值，並返回只包含唯一元素的數組。這個函數需要一個數組，它可以是未排序的，作為它的唯一參數。

例如，以下inputArray內容包含一系列重複值：

```
{"inputArray": [1,2,3,3,3,3,3,3,4] }
```

您可以使用該States.ArrayUnique函數作為並指定要從中刪除重複值的數組：

```
"array.$": "States.ArrayUnique($.inputArray)"
```

該States.ArrayUnique函數將返回以下僅包含唯一元素的數組，刪除所有重複值：

```
{"array": [1,2,3,4] }
```

數據編碼和解碼的內在函數

使用下列內建函數根據 Base64 編碼配置編碼或解碼資料。

States.Base64Encode

使用States.Base64Encode內建函數根據 MIME Base64 編碼配置來編碼資料。您可以使用此函數將數據傳遞給其他AWS服務，而無需使用AWS Lambda函數。

此函數需要最多 10,000 個字元的資料字串來編碼為唯一引數。

例如，請考慮下列input字串：

```
{"input": "Data to encode" }
```

您可以使用該States.Base64Encode函數將input字符串編碼為 MIME Base64 字符串：

```
"base64.$": "States.Base64Encode($.input)"
```

該States.Base64Encode函數返回以下編碼數據作為響應：

```
{"base64": "RGF0YSB0byBlbmNvZGU=" }
```

States.Base64Decode

使用States.Base64Decode內建函數根據 MIME Base64 解碼配置來解碼資料。您可以使用此函數將資料傳遞至其他AWS服務，而不需使用 Lambda 函數。

此函數需要一個最多 10,000 個字符的 Base64 編碼數據字符串作為其唯一的參數進行解碼。

例如，假設有下列輸入：

```
{"base64": "RGF0YSB0byBlbmNvZGU=" }
```

您可以使用該States.Base64Decode函數將 base64 字符串解碼為人類可讀的字符串：

```
"data.$": "States.Base64Decode($.base64)"
```

States.Base64Decode function將返回以下解碼數據作為響應：

```
{"data": "Decoded data" }
```

散列計算的內在

States.Hash

使用States.Hash內建函數來計算給定輸入的雜湊值。您可以使用此函數將資料傳遞至其他AWS服務，而不需使用 Lambda 函數。

這個函數有兩個參數。第一個參數是要計算散列值的數據。第二個引數是用來執行雜湊計算的雜湊演算法。您提供的資料必須是包含 10,000 個字元或更少的物件字串。

您指定的雜湊演算法可以是下列任一演算法：

- MD5
- SHA-1
- SHA-256
- SHA-384
- SHA-512

例如，您可以使用此函數來計算使用指定的Data字符串的哈希值Algorithm：

```
{
   "Data": "input data",
   "Algorithm": "SHA-1"
}
```

您可以使用States.Hash函數來計算哈希值：

```
"output.$": "States.Hash($.Data, $.Algorithm)"
```

該States.Hash函數返回響應以下哈希值：

```
{"output": "aaff4a450a104cd177d28d18d7485e8cae074b7" }
```

JSON 數據操作的內在函數

您可以使用這些函數對 JSON 物件執行基本的資料處理作業。

States.JsonMerge

使用States.JsonMerge內建函式將兩個 JSON 物件合併為單一物件。這個函數有三個參數。前兩個參數是您要合併的 JSON 物件。第三個引數是的布林值false。此布林值決定是否啟用深度合併模式。

目前，Step Functions 僅支援淺層合併模式；因此，您必須將布林值指定為false。在淺模式下，如果兩個 JSON 對象中都存在相同的密鑰，則後一個對象的密鑰將覆蓋第一個對象中的相同密鑰。此外，當您使用淺層合併時，不會合併 JSON 物件中巢狀的物件。

例如，您可以使用States.JsonMerge函數來合併下列共用金鑰的 JSON 物件a。

```
{
   "json1": { "a": {"a1": 1, "a2": 2}, "b": 2 },
   "json2": { "a": {"a3": 1, "a4": 2}, "c": 3 }
}
```

您可以將 json1 和 json2 對象指定為States.JsonMerge函數中的輸入以將它們合併在一起：

```
"output.$": "States.JsonMerge($.json1, $.json2, false)"
```

會States.JsonMerge傳回下列合併的 JSON 物件作為結果。在合併的 JSON 物件中output，物json2件的索引鍵a會取代json1物件的索引鍵a。另外，由於淺模式不支持合併嵌套json1對象，因此丟棄對象鍵a中的嵌套對象。

```
{
   "output": {
      "a": {"a3": 1, "a4": 2},
      "b": 2,
      "c": 3
   }
}
```

States.StringToJson

該States.StringToJson函數將轉義 JSON 字符串的引用路徑作為其唯一參數。

解釋器應用 JSON 解析器，並返回輸入的解析 JSON 形式。例如，您可以使用此函數來逸出下列輸入字串：

```
{
 "escapedJsonString": "{\"foo\": \"bar\"}"
}
```

使用States.StringToJson函數並指定escapedJsonString為輸入引數：

```
States.StringToJson($.escapedJsonString)
```

該States.StringToJson函數返回以下結果：

```
{ "foo": "bar" }
```

States.JsonToString

該States.JsonToString函數只需要一個參數，這是包含要作為未轉義字符串返回的 JSON 數據的路徑。解釋器返回一個字符串，其中包含表示由路徑指定的數據的 JSON 文本。例如，您可以提供下列包含逸出值的 JSON 路徑：

```
{
  "unescapedJson": {
     "foo": "bar"
  }
}
```

為States.JsonToString函數提供包含在其中的數據unescapedJson：

```
States.JsonToString($.unescapedJson)
```

該States.JsonToString函數返回以下響應：

```
{\"foo\": \"bar\"}
```

數學運算的內在函數

您可以使用這些函數來執行數學運算。

States.MathRandom

使用States.MathRandom內在函數返回指定的起始編號（包括）和結束編號（排斥）之間的隨機數。

您可以使用此函數在兩個或多個資源之間分配特定任務。

這個函數有三個參數。第一個參數是開始編號，第二個參數是結束編號，最後一個參數控制種子值。種子值參數是可選的。如果你使用這個函數與相同的種子值，它返回一個相同的數字。

> **Important**
>
> 由於States.MathRandom函數不會傳回密碼編譯安全隨機數，因此建議您不要將它用於安全性敏感的應用程式。

輸入驗證

- 您必須為起始編號和結束編號引數指定整數值。

  如果您為起始編號或結束編號引數指定非整數值，Step Functions 會將其四捨五入為最接近的整數。

例如，若要產生介於 1 到 999 之間的隨機數字，您可以使用下列輸入值：

```
{
   "start": 1,
   "end": 999
}
```

若要產生隨機數，請將 `start` and `end` 值提供給`States.MathRandom`函數：

```
"random.$": "States.MathRandom($.start, $.end)"
```

該`States.MathRandom`函數返回以下隨機數作為響應：

```
{"random": 456 }
```

States.MathAdd

使用`States.MathAdd`內在函數返回兩個數字的總和。例如，您可以使用此函數在迴圈內增加值，而不叫用 Lambda 函數。

輸入驗證

- 您必須為所有引數指定整數值。

  如果您為一個或兩個引數指定非整數值，Step Functions 會將其四捨五入為最接近的整數。
- 您必須在 -2147483648 和 2147483647 的範圍內指定整數值。

例如，您可以使用下列值從 111 減去一個值：

```
{
   "value1": 111,
   "step": -1
}
```

然後，使用States.MathAdd函數定義value1為起始值，並step作為遞增value1的值：

```
"value1.$": "States.MathAdd($.value1, $.step)"
```

該States.MathAdd函數將返回以下數字作為響應：

```
{"value1": 110 }
```

字符串操作的內在

States.StringSplit

使用States.StringSplit內建函式將字串分割成值陣列。這個函數有兩個參數。第一個參數是一個字符串，第二個參數是分隔字符，該函數將用於分割字符串。

Example -使用單一分隔字元分割輸入字串

在此範例中，使用States.StringSplit來除以下項目inputString，其中包含一系列逗號分隔值：

```
{
    "inputString": "1,2,3,4,5",
    "splitter": ","
}
```

使用States.StringSplit函數並定義inputString為第一個引數，並將分隔字元splitter作為第二個引數：

```
"array.$": "States.StringSplit($.inputString, $.splitter)"
```

該States.StringSplit函數返回以下字符串數組作為結果：

```
{"array":  ["1","2","3","4","5"] }
```

Example -使用多個分隔字元分割輸入字串

在此範例中，使用States.StringSplit來除以下項目inputString，其中包含多個分隔字元：

```
{
  "inputString": "This.is+a,test=string",
```

```
  "splitter": ".+,="
}
```

使用States.StringSplit函數，如下所示：

```
{
  "myStringArray.$": "States.StringSplit($.inputString, $.splitter)"
}
```

該States.StringSplit函數返回以下字符串數組作為結果：

```
{"myStringArray": [
  "This",
  "is",
  "a",
  "test",
  "string"
]}
```

唯一標識符生成的內在

States.UUID

使用States.UUID內部函數返回使用隨機數生成的版本 4 通用唯一標識符（v4 UUID）。例如，您可以使用此函數呼叫需要 UUID 參數的其他AWS服務或資源，或在 DynamoDB 表格中插入項目。

該States.UUID函數被調用，沒有指定參數：

```
"uuid.$": "States.UUID()"
```

函數會傳回隨機產生的 UUID，如下列範例所示：

```
{"uuid": "ca4c1140-dcc1-40cd-ad05-7b4aa23df4a8" }
```

泛型操作的內在

States.Format

使用States.Format內建函式，從常值和內插值建構字串。這個函數需要一個或多個參數。第一個參數的值必須是一個字符串，並且可以包含字符序列的零個或多個實例{}。內在的調用中必須有

盡可能多的剩餘參數，因為存在. {} 解釋器返回第一個參數中定義的字符串，每個參數{}由內在調用中的位置對應參數的值替換。

例如，您可以使用以下個人的輸入name，以及將其名稱插入到一個template句子中：

```
{
 "name": "Arnav",
 "template": "Hello, my name is {}."
}
```

使用該States.Format函數並指定template要插入的字符串和字符串代替{}字符：

```
States.Format('Hello, my name is {}.', $.name)
```

或

```
States.Format($.template, $.name)
```

對於以前的任一輸入，該States.Format函數返回完成的字符串作為響應：

```
Hello, my name is Arnav.
```

內在函數中的保留字符

下列字元是為內建函數保留的，如果您希望這些字元出現在 Value: ' { }、和，則必須使用反斜線 ('\') 逸出。\

如果字元\需要顯示為值的一部分，而不用作逸出字元，則必須使用反斜線將其逸出。下面的轉義字符序列與內在函數一起使用：

- 文字字串\'代表'。
- 文字字串\{代表{。
- 文字字串\}代表}。
- 文字字串\\代表\。

在 JSON 中，字串常值中包含的反斜線必須以另一個反斜線逸出。JSON 的等效列表是：

- 轉義字符串\\\'表示\'。

- 轉義字符串\\\{表示\{。
- 轉義字符串\\\}表示\}。
- 轉義字符串\\\\表示\\。

> **Note**
>
> 如果在內部調用字符串中找到\一個打開的轉義反斜杠，解釋器將返回一個運行時錯誤。

常見狀態欄位

`Type` (必要)

狀態的類型。

`Next`

下一個狀態的名稱將會在目前狀態完成時執行。有些狀態類型 (例如 `Choice`) 允許使用多個轉移狀態。

如果目前狀態是工作流程中的最後一個狀態或終端機狀態 (例如Succeed或)Fail，則不需要指定`Next`欄位。

`End`

如果設定為 `true`，請將此狀態指定為結束狀態 (結束執行)。每個狀態機器可以有任意數量的結束狀態。只有其中一個 `Next` 或 `End` 可使用於狀態中。某些狀態類型，例如`Choice`，或終端狀態，例如Succeed和Fail，不支援或使用End欄位。

`Comment` (選用)

保留人類可讀的狀態描述。

`InputPath` (選用)

一個路徑，可供選取一部分的狀態輸入以傳遞至狀態的任務進行處理。如果省略，它具有可指定整個輸入的 $ 值。如需詳細資訊，請參閱輸入和輸出處理。

`OutputPath` (選用)

選取要傳遞至下一個狀態之狀態輸出部分的路徑。如果省略，它具有指$定整個輸出的值。如需詳細資訊，請參閱輸入和輸出處理。

Pass

Pass 狀態 ("Type": "Pass") 會將其輸入傳遞到其輸出，而不執行任何工作。在建構及偵錯狀態機器時，Pass 狀態非常實用。

您也可以使用Pass狀態來使用篩選器轉換 JSON 狀態輸入，然後將轉換後的資料傳遞至工作流程中的下一個狀態。如需有關輸入轉換的資訊，請參閱InputPath、參數和 ResultSelector。

除了常見狀態欄位以外，Pass 狀態還允許使用下列欄位。

Result (選用)

指傳遞到下一個狀態的虛擬任務的輸出。如果您在狀態機定義中包含ResultPath欄位，Result則會依照指定的方式放置ResultPath並傳遞至下一個狀態。

ResultPath (選用)

指定在中指定之虛擬工作的輸出 (相對於輸入) 的放置位置Result。輸入會先依據 OutputPath 欄位 (如果有的話) 所指定進一步篩選，而後做為狀態的輸出。如需詳細資訊，請參閱輸入和輸出處理。

Parameters (選用)

創建將作為輸入傳遞的鍵-值對的集合。您可以指定Parameters為靜態值，也可以使用路徑從輸入中進行選取。如需詳細資訊，請參閱InputPath、參數和 ResultSelector。

Pass 狀態範例

以下是 Pass 狀態的範例，該範例會將一些固定資料插入狀態機器 (可能用於進行測試)。

```
"No-op": {
  "Type": "Pass",
  "Result": {
    "x-datum": 0.381018,
    "y-datum": 622.2269926397355
  },
  "ResultPath": "$.coords",
  "End": true
}
```

假設此狀態的輸入如下。

```
{
  "georefOf": "Home"
}
```

然後，輸出會是這個。

```
{
  "georefOf": "Home",
  "coords": {
    "x-datum": 0.381018,
    "y-datum": 622.2269926397355
  }
}
```

任務

Task 狀態 ("Type": "Task") 代表狀態機器執行的一個工作單位。任務通過使用活動或AWS Lambda函數來執行工作，通過與其他支持的集成AWS 服務，或者通過調用第三方 API（例如 Stripe）來執行工作。

Amazon 狀態語言代表任務，方法是將某個州的類型設定為，Task並將任務提供活動的 Amazon 資源名稱 (ARN)、Lambda 函數或第三方 API 端點。下列工作狀態定義會叫用名為*HelloFunction*的 Lambda 函數。

```
"Lambda Invoke": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "Parameters": {
    "Payload.$": "$",
    "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:HelloFunction:
$LATEST"
  },
  "End": true
}
```

在本主題中

- 工作類型
- 工作狀態欄位
- 工作狀態定義範例

- 活動

工作類型

「Step Functions」支援您可在「工作」狀態定義中指定的下列工作類型：

- 活動
- Lambda 函數
- 一個支持 AWS 服務
- 一個 HTTP 任務

您可以在任務狀態定義的Resource欄位中提供其 ARN 來指定任務類型。下列範例顯示Resource欄位的語法。除了調用第三方 API 的任務類型之外的所有任務類型，都使用以下語法。如需 HTTP 工作語法的相關資訊，請參閱呼叫第三方 API。

在「任務」狀態定義中，將下列語法中的斜體文字取代為AWS資源特定的資訊。

```
arn:partition:service:region:account:task_type:name
```

下列清單說明此語法中的個別元件：

- partition 是要使用的 AWS Step Functions 分割區，最常使用 aws。
- service表示AWS 服務用來執行工作，且可以是下列其中一個值：
  - states對於一項活動。
  - lambda對於一個 Lambda 函數。如果您與其他 (例如AWS 服務，Amazon SNS 或亞馬遜 DynamoDB) 整合，請使用sns或。dynamodb
- region是已建立 Step Functions 數活動或狀態機器類型、Lambda 函數或任何其他AWS資源的AWS區域代碼。
- account是您在其中定義資源的 AWS 帳戶 ID。
- task_type 是要執行的任務類型。它可能是以下其中一個數值：
  - activity— 一項活動。
  - function-一個 Lambda 函數。
  - *servicename*— 支援的連線服務的名稱 (請參閱Step Functions 的最佳化整合)。
- name是已註冊的資源名稱 (活動名稱、Lambda 函數名稱或服務 API 動作)。

> **Note**
>
> Step Functions 不支援跨分割區或區域參考 ARN。例如，不`aws-cn`能調用`aws`分區中的任務，反之亦然。

下列各節會提供每個任務類型的詳細資訊。

活動

活動代表由您實作和託管並可執行特定任務的工作者 (程序或執行緒)。活動僅受到標準工作流程支援，不受快速工作流程支持。

活動 `Resource` ARN 會使用以下語法。

```
arn:partition:states:region:account:activity:name
```

> **Note**
>
> 您必須先建立具有 Step Functions 的活動 (使用CreateActivity、API 動作或 Step Functions 主控台)，才能首次使用步驟函數。

如需建立活動及實作工作者的詳細資訊，請參閱活動。

Lambda 函數

Lambda 工作會使用AWS Lambda. 若要指定 Lambda 函數，請在`Resource`欄位中使用 Lambda 函數的 ARN。

根據您用來指定 Lambda 函數的整合類型 (最佳化整合或 AWSSDK 整合)，Lambda 函數`Resource`欄位的語法會有所不同。

下列`Resource`欄位語法是與 Lambda 函數進行最佳化整合的範例。

```
"arn:aws:states:::lambda:invoke"
```

下列`Resource`欄位語法是 AWS SDK 與 Lambda 函數整合的範例。

```
"arn:aws:states:::aws-sdk:lambda:invoke"
```

下列Task狀態定義顯示與名為的 Lambda 函數進行最佳化整合的範例*HelloWorld*。

```
"LambdaState": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "OutputPath": "$.Payload",
  "Parameters": {
    "Payload.$": "$",
    "FunctionName": "arn:aws:lambda:us-east-1:function:HelloWorld:$LATEST"
  },
  "Next": "NextState"
}
```

在Resource欄位中指定的 Lambda 函數完成後，其輸出會傳送至Next欄位 (」NextState「) 中識別的狀態。

一個支持 AWS 服務

當您參考連線的資源時，Step Functions 會直接呼叫支援服務的 API 動作。在 Resource 欄位中指定服務和動作。

已連線的服務 Resource ARN 會使用以下語法。

```
arn:partition:states:region:account:servicename:APIname
```

> **Note**
>
> 若要對已連線的資源建立同步連線，請將 .sync 附加到 ARN 中的 *APIname* 項目。如需詳細資訊，請參閱使用其他 服務。

例如：

```
{
 "StartAt": "BATCH_JOB",
 "States": {
   "BATCH_JOB": {
     "Type": "Task",
     "Resource": "arn:aws:states:::batch:submitJob.sync",
     "Parameters": {
       "JobDefinition": "preprocessing",
       "JobName": "PreprocessingBatchJob",
```

```
        "JobQueue": "SecondaryQueue",
        "Parameters.$": "$.batchjob.parameters",
        "RetryStrategy": {
           "attempts": 5
         }
      },
      "End": true
    }
  }
}
```

工作狀態欄位

除了常見狀態欄位以外，Task 狀態還有下列欄位。

`Resource` (必要)

URI，特別是能夠唯一識別要執行特定任務的 ARN。

`Parameters` (選用)

用來將資訊傳遞給已連線資源的 API 動作。這些參數可以混合使用靜態 JSON 和 JsonPath. 如需詳細資訊，請參閱將參數傳遞至服務 API。

`Credentials` (選用)

指定狀態機器的執行角色必須承擔的目標角色，然後再呼叫指Resource定的角色。或者，您也可以指定 JsonPath 值或內建函數，在執行階段根據執行輸入解析為 IAM 角色 ARN。如果您指定 JSONPath 值，則必須在$.其前面加上符號。

如需在Task狀態中使用此欄位的範例，請參閱作業狀態的證明資料欄位範例。如需使用此欄位從狀態機存取跨帳號AWS資源的範例，請參閱教學課程：存取跨帳戶 AWS 資源。

> **Note**
>
> 使用 Lambda 函數和支援AWS服務的欄位支援此欄位。工作類型

`ResultPath` (選用)

指定要將執行 Resource 中所指定任務的結果放置於何處 (在輸入中)。輸入會先依據 OutputPath 欄位 (如果有的話) 所指定篩選，而後做為狀態的輸出。如需詳細資訊，請參閱輸入和輸出處理。

`ResultSelector` (選用)

傳遞鍵值對的集合，其中值是靜態的或從結果中選擇的。如需詳細資訊，請參閱ResultSelector。

`Retry` (選用)

稱為 Retrier 的物件陣列，這類物件可定義狀態發生執行時間錯誤時的重試政策。如需詳細資訊，請參閱使用重試和使用 Catch 的狀態機器示例。

`Catch` (選用)

稱為 Catcher 的物件陣列，可定義後援狀態。如果狀態遇到執行時間錯誤並且其重試政策耗盡或未定義，則執行此狀態。如需詳細資訊，請參閱備用狀態。

`TimeoutSeconds` (選用)

指定活動或工作在因States.Timeout錯誤而失敗而逾時之前可以執行的時間上限。逾時值必須是正的非零整數。預設值為 99999999。

逾時計數會在工作啟LambdaFunctionStarted動後開始，例如，在執行事件歷程記錄中記錄ActivityStarted或事件時。對於活動，計數會在GetActivityTask收到權杖時開始計數，並ActivityStarted記錄在執行事件歷程記錄中。

當任務啟動時，Step Functions 會在指定的TimeoutSeconds持續時間內等待任務或活動 Worker 的成功或失敗回應。如果任務或活動 Worker 無法在此時間內回應，「Step Functions」會將工作流程執行標記為失敗。

`TimeoutSecondsPath` (選用)

如果您想要使用參考路徑從狀態輸入動態提供逾時值，請使用TimeoutSecondsPath。解析後，參考路徑必須選取值為正整數的欄位。

> **Note**
>
> 狀Task態不能同時包含TimeoutSeconds和TimeoutSecondsPath。

`HeartbeatSeconds` (選用)

決定活動 Worker 在執行任務期間傳送的活動訊號頻率。活動訊號表示工作仍在執行中，需要更多時間才能完成。活動訊號可防止活動或工作在持續時TimeoutSeconds間內逾時。

HeartbeatSeconds必須是小於欄位值的正非零整數TimeoutSeconds值。預設值為99999999。如果工作的活動訊號之間經過的時間超過指定秒數，則工作狀態會失敗並顯示錯誤。States.Timeout

對於活動，計數會在GetActivityTask收到權杖時開始計數，並ActivityStarted記錄在執行事件歷程記錄中。

HeartbeatSecondsPath (選用)

如果您想要使用參考路徑從狀態輸入動態提供活動訊號值，請使用HeartbeatSecondsPath。解析後，參考路徑必須選取值為正整數的欄位。

> **Note**
>
> 狀Task態不能同時包含HeartbeatSeconds和HeartbeatSecondsPath。

如果狀態結束執行，則 Task 狀態必須將 End 欄位設定為 true，或必須在 Task 狀態完成時於 Next 欄位中提供執行的狀態。

工作狀態定義範例

下列範例顯示如何根據您的需求指定 Task 狀態定義。

- 指定工作狀態逾時和活動訊號間隔
  - 靜態逾時和活動訊號通知範例
  - 動態工作逾時和活動訊號通知範例
- 使用認證欄位
  - 指定硬式編碼的 IAM 角色 ARN
  - 將 JSON 路徑指定為 IAM 角色 ARN
  - 將內建函數指定為 IAM 角色 ARN

工作狀態逾時和活動訊號間隔

這是針對長時間執行的活動設定逾時值和活動訊號間隔的最佳實務。這可以透過指定逾時值和活動訊號值，或透過動態設定來完成。

靜態逾時和活動訊號通知範例

當 HelloWorld 完成時，將會執行下一個狀態 (這裡稱為 NextState)。

如果這個任務無法在 300 秒內完成，或者未在 60 秒的間隔內傳送活動訊號通知，則任務會被標示為 `failed`。

```
"ActivityState": {
  "Type": "Task",
  "Resource": "arn:aws:states:us-east-1:123456789012:activity:HelloWorld",
  "TimeoutSeconds": 300,
  "HeartbeatSeconds": 60,
  "Next": "NextState"
}
```

動態工作逾時和活動訊號通知範例

在此範例中，當AWS Glue工作完成時，將會執行下一個狀態。

如果此工作無法在工作動態設定的間隔內完成，則會將AWS Glue工作標記為`failed`。

```
"GlueJobTask": {
  "Type": "Task",
  "Resource": "arn:aws:states:::glue:startJobRun.sync",
  "Parameters": {
    "JobName": "myGlueJob"
  },
  "TimeoutSecondsPath": "$.params.maxTime",

  "Next": "NextState"
}
```

作業狀態的證明資料欄位範例

指定硬式編碼的 IAM 角色 ARN

下列範例會指定目標 IAM 角色，狀態機器的執行角色必須假設該角色才能存取名為Echo的跨帳戶 Lambda 函數。在此範例中，目標角色 ARN 會指定為硬式編碼值。

```
{
  "StartAt": "Cross-account call",
  "States": {
    "Cross-account call": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
```

```
      "Credentials": {
        "RoleArn": "arn:aws:iam::111122223333:role/LambdaRole"
      },
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:Echo"
      },
      "End": true
    }
  }
}
```

將 JSON 路徑指定為 IAM 角色 ARN

下列範例會指定 JSONPath 值，這個值會在執行階段解析為 IAM 角色 ARN。

```
{
  "StartAt": "Lambda",
  "States": {
    "Lambda": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn.$": "$.roleArn"
      },
      ...
    }
  }
}
```

將內建函數指定為 IAM 角色 ARN

下列範例會使用States.Format內建函式，在執行階段解析為 IAM 角色 ARN。

```
{
  "StartAt": "Lambda",
  "States": {
    "Lambda": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn.$": "States.Format('arn:aws:iam::{}:role/ROLENAME', $.accountId)"
      },
      ...
```

```
    }
  }
}
```

活動

活動是一項AWS Step Functions功能，可讓您在狀態機器中執行工作，工作是由可在 Amazon 彈性運算雲端 (Amazon EC2)、Amazon 彈性容器服務 (Amazon ECS)、行動裝置上託管的工作者執行工作 — 基本上在任何地方。

概要

在 AWS Step Functions 中，活動可將在某處執行的程式碼 (也稱為活動工作者) 與狀態機器中的特定任務建立關聯。您可以使用 Step Functions 主控台或呼叫來建立活動`CreateActivity`。這會為您的任務狀態提供亞馬遜資源名稱 (ARN)。使用此 ARN 來輪詢任務狀態，以便在您的活動工作者中使用。

> **Note**
>
> 活動並未進行版本控制，應該可與舊版相容。如果您必須對活動進行向後不相容的變更，請使用唯一名稱在「步驟函數」中建立新活動。

活動工作者可以是在 Amazon EC2 執行個體上執行的應用程式、AWS Lambda函數、行動裝置：任何可以建立 HTTP 連線的應用程式，託管在任何地方。當「步驟函數」達到活動任務狀態時，工作流程會等待活動 Worker 輪詢任務。活動工作者通過使用`GetActivityTask`，並發送相關活動的 ARN 輪詢步驟函數。 `GetActivityTask`傳回回應，其中包括 `input` (工作的 JSON 輸入字串) 和 `taskToken`(工作的唯一識別碼)。在活動工作者完成其工作之後，就可以使用 `SendTaskSuccess` 或 `SendTaskFailure` 來提供成功或失敗的報告。這兩個呼叫會使用 `GetActivityTask` 所提供的 `taskToken`，將結果與該任務建立關聯。

活動任務的相關 API

Step Functions 提供用於建立和列出活動、要求工作，以及根據 Worker 的結果管理狀態機器流程的 API。

以下是與活動相關的步驟函數 API：

- `CreateActivity`
- `GetActivityTask`

- ListActivities
- SendTaskFailure
- SendTaskHeartbeat
- SendTaskSuccess

> **Note**
>
> 透過 GetActivityTask 輪詢活動任務，可能會導致某些實作延遲。請參閱 輪詢活動任務時避免延遲。

等待活動任務完成

在任務定義中設定 TimeoutSeconds，以設定狀態等待的時間長度。若要讓任務保持作用中並且等待，請在 TimeoutSeconds 中設定的時間內使用 SendTaskHeartbeat，定期從您的活動工作者傳送活動訊號。藉由設定較長的逾時持續時間並主動傳送活動訊號，Step Functions 中的活動最多可等待一年的執行完成。

例如，如果您需要一個可等待冗長程序結果的工作流程，請執行下列動作：

1. 使用主控台來建立活動，或藉由呼叫 CreateActivity 來建立它。記下活動 ARN。
2. 在狀態機器定義的活動任務狀態中參考該 ARN，並且設定 TimeoutSeconds。
3. 使用 GetActivityTask，並且參考該 ARN，以實作可輪詢工作的活動工作者。
4. 在您於狀態機器任務定義的 HeartbeatSeconds 中所設定的時間內，定期使用 SendTaskHeartbeat，讓任務不會逾時。
5. 開始執行您的狀態機器。
6. 啟動您的活動工作者程序。

執行會暫停於活動任務狀態，並等待活動工作者輪詢任務。將 taskToken 提供給活動工作者後，工作流程就會等待 SendTaskSuccess 或 SendTaskFailure 提供狀態。如果在 TimeoutSeconds 中設定的時間前，執行並未收到這些資訊或 SendTaskHeartbeat 呼叫，則執行會失敗且執行歷史記錄中會包含 ExecutionTimedOut 事件。

後續步驟

如要詳細了解建立使用活動工作者的狀態機器，請參閱：

- 使用 Step Functions 創建活動狀態機
- Ruby 中的範例活動工作者

Ruby 中的範例活動工作者

以下是範例活動工作者，其使用 AWS SDK for Ruby 來示範如何使用最佳實務及實作自己的活動工作者。

此程式碼會實作可為輪詢器和活動工作者設定執行緒數量的消費者-生產者模式。輪詢器執行緒會不斷地輪詢活動任務。擷取活動任務後，它就會通過限制封鎖佇列，以便活動執行緒加以挑選。

- 如需有關的詳細資訊AWS SDK for Ruby，請參閱 AWS SDK for RubyAPI 參考。
- 若要下載此程式碼和相關資源，請參閱上GitHub的 step-functions-ruby-activity-worker 儲存庫。

以下 Ruby 程式碼是這個範例 Ruby 活動工作者的主要進入點。

```
require_relative '../lib/step_functions/activity'
credentials = Aws::SharedCredentials.new
region = 'us-west-2'
activity_arn = 'ACTIVITY_ARN'

activity = StepFunctions::Activity.new(
  credentials: credentials,
  region: region,
  activity_arn: activity_arn,
  workers_count: 1,
  pollers_count: 1,
  heartbeat_delay: 30
)

# The start method takes as argument the block that is the actual logic of your custom
activity.
activity.start do |input|
  { result: :SUCCESS, echo: input['value'] }
end
```

此程式碼包含您可以變更的預設值，以便參考您的活動，以及針對您的特定實作加以調整。此程式碼會將實際實作邏輯當作輸入，讓您參考特定活動和認證，以及讓您設定執行緒數目和活動訊號延遲。如需詳細資訊並下載程式碼，請參閱步驟函式 Ruby 活動背景工作者。

項目	描述
require_relative	以下範例活動工作者程式碼的相對路徑。
region	活動的 AWS 區域。
workers_count	您的活動工作者的執行緒數目。對於大多數實作，10 到 20 個執行緒應該就足夠。處理活動所需的時間越長，就可能需要更多執行緒。預估時，以每秒處理的活動數目乘以第 99 個百分位數活動處理延遲 (以秒為單位)。
pollers_count	您的輪詢器的執行緒數目。對於大多數實作，10 到 20 個執行緒應該就足夠。
heartbeat_delay	活動訊號之間的延遲 (以秒為單位)。
input	實作您的活動邏輯。

以下是 Ruby 活動工作者，其參考您程式碼中的 `../lib/step_functions/activity`。

```
require 'set'
require 'json'
require 'thread'
require 'logger'
require 'aws-sdk'

module Validate
  def self.positive(value)
    raise ArgumentError, 'Argument has to be positive' if value <= 0
    value
  end

  def self.required(value)
    raise ArgumentError, 'Argument is required' if value.nil?
    value
  end
end
```

```
module StepFunctions
  class RetryError < StandardError
    def initialize(message)
      super(message)
    end
  end

  def self.with_retries(options = {}, &block)
    retries = 0
    base_delay_seconds = options[:base_delay_seconds] || 2
    max_retries = options[:max_retries] || 3
    begin
      block.call
    rescue => e
      puts e
      if retries < max_retries
        retries += 1
        sleep base_delay_seconds**retries
        retry
      end
      raise RetryError, 'All retries of operation had failed'
    end
  end

  class Activity
    def initialize(options = {})
      @states = Aws::States::Client.new(
        credentials: Validate.required(options[:credentials]),
        region: Validate.required(options[:region]),
        http_read_timeout: Validate.positive(options[:http_read_timeout] || 60)
      )
      @activity_arn = Validate.required(options[:activity_arn])
      @heartbeat_delay = Validate.positive(options[:heartbeat_delay] || 60)
      @queue_max = Validate.positive(options[:queue_max] || 5)
      @pollers_count = Validate.positive(options[:pollers_count] || 1)
      @workers_count = Validate.positive(options[:workers_count] || 1)
      @max_retry = Validate.positive(options[:workers_count] || 3)
      @logger = Logger.new(STDOUT)
    end

    def start(&block)
      @sink = SizedQueue.new(@queue_max)
      @activities = Set.new
      start_heartbeat_worker(@activities)
```

```
      start_workers(@activities, block, @sink)
      start_pollers(@activities, @sink)
      wait
    end

    def queue_size
      return 0 if @sink.nil?
      @sink.size
    end

    def activities_count
      return 0 if @activities.nil?
      @activities.size
    end

    private

    def start_pollers(activities, sink)
      @pollers = Array.new(@pollers_count) do
        PollerWorker.new(
          states: @states,
          activity_arn: @activity_arn,
          sink: sink,
          activities: activities,
          max_retry: @max_retry
        )
      end
      @pollers.each(&:start)
    end

    def start_workers(activities, block, sink)
      @workers = Array.new(@workers_count) do
        ActivityWorker.new(
          states: @states,
          block: block,
          sink: sink,
          activities: activities,
          max_retry: @max_retry
        )
      end
      @workers.each(&:start)
    end

    def start_heartbeat_worker(activities)
```

```
      @heartbeat_worker = HeartbeatWorker.new(
        states: @states,
        activities: activities,
        heartbeat_delay: @heartbeat_delay,
        max_retry: @max_retry
      )
      @heartbeat_worker.start
    end

    def wait
      sleep
    rescue Interrupt
      shutdown
    ensure
      Thread.current.exit
    end

    def shutdown
      stop_workers(@pollers)
      wait_workers(@pollers)
      wait_activities_drained
      stop_workers(@workers)
      wait_activities_completed
      shutdown_workers(@workers)
      shutdown_worker(@heartbeat_worker)
    end

    def shutdown_workers(workers)
      workers.each do |worker|
        shutdown_worker(worker)
      end
    end

    def shutdown_worker(worker)
      worker.kill
    end

    def wait_workers(workers)
      workers.each(&:wait)
    end

    def wait_activities_drained
      wait_condition { @sink.empty? }
    end
```

```
def wait_activities_completed
  wait_condition { @activities.empty? }
end

def wait_condition(&block)
  loop do
    break if block.call
    sleep(1)
  end
end

def stop_workers(workers)
  workers.each(&:stop)
end

class Worker
  def initialize
    @logger = Logger.new(STDOUT)
    @running = false
  end

  def run
    raise 'Method run hasn\'t been implemented'
  end

  def process
    loop do
      begin
        break unless @running
        run
      rescue => e
        puts e
        @logger.error('Unexpected error has occurred')
        @logger.error(e)
      end
    end
  end

  def start
    return unless @thread.nil?
    @running = true
    @thread = Thread.new do
      process
```

```
        end
      end

      def stop
        @running = false
      end

      def kill
        return if @thread.nil?
        @thread.kill
        @thread = nil
      end

      def wait
        @thread.join
      end
    end

    class PollerWorker < Worker
      def initialize(options = {})
        @states = options[:states]
        @activity_arn = options[:activity_arn]
        @sink = options[:sink]
        @activities = options[:activities]
        @max_retry = options[:max_retry]
        @logger = Logger.new(STDOUT)
      end

      def run
        activity_task = StepFunctions.with_retries(max_retry: @max_retry) do
          begin
            @states.get_activity_task(activity_arn: @activity_arn)
          rescue => e
            @logger.error('Failed to retrieve activity task')
            @logger.error(e)
          end
        end
        return if activity_task.nil? || activity_task.task_token.nil?
        @activities.add(activity_task.task_token)
        @sink.push(activity_task)
      end
    end

    class ActivityWorker < Worker
```

```
    def initialize(options = {})
      @states = options[:states]
      @block = options[:block]
      @sink = options[:sink]
      @activities = options[:activities]
      @max_retry = options[:max_retry]
      @logger = Logger.new(STDOUT)
    end

    def run
      activity_task = @sink.pop
      result = @block.call(JSON.parse(activity_task.input))
      send_task_success(activity_task, result)
    rescue => e
      send_task_failure(activity_task, e)
    ensure
      @activities.delete(activity_task.task_token) unless activity_task.nil?
    end

    def send_task_success(activity_task, result)
      StepFunctions.with_retries(max_retry: @max_retry) do
        begin
          @states.send_task_success(
            task_token: activity_task.task_token,
            output: JSON.dump(result)
          )
        rescue => e
          @logger.error('Failed to send task success')
          @logger.error(e)
        end
      end
    end

    def send_task_failure(activity_task, error)
      StepFunctions.with_retries do
        begin
          @states.send_task_failure(
            task_token: activity_task.task_token,
            cause: error.message
          )
        rescue => e
          @logger.error('Failed to send task failure')
          @logger.error(e)
        end
```

```
        end
      end
    end

    class HeartbeatWorker < Worker
      def initialize(options = {})
        @states = options[:states]
        @activities = options[:activities]
        @heartbeat_delay = options[:heartbeat_delay]
        @max_retry = options[:max_retry]
        @logger = Logger.new(STDOUT)
      end

      def run
        sleep(@heartbeat_delay)
        @activities.each do |token|
          send_heartbeat(token)
        end
      end

      def send_heartbeat(token)
        StepFunctions.with_retries(max_retry: @max_retry) do
          begin
            @states.send_task_heartbeat(token)
          rescue => e
            @logger.error('Failed to send heartbeat for activity')
            @logger.error(e)
          end
        end
      rescue => e
        @logger.error('Failed to send heartbeat for activity')
        @logger.error(e)
      end
    end
  end
end
```

Choice

狀Choice態 ("Type": "Choice") 將條件邏輯添加到狀態機。

除了大多數常見的狀態欄位之外，Choice狀態還包含下列其他欄位。

Choices (必要)

選擇規則陣列，該陣列可決定狀態機器接下來會轉換到哪個狀態。您可以在「選擇規則」中使用比較運算子來比較輸入變數與特定值。例如，使用「選擇規則」，您可以比較輸入變數是否大於或小於 100。

執行Choice狀態時，它會將每個選擇規則評估為 true 或 false。根據此評估的結果，「步驟函數」會轉換至工作流程中的下一個狀態。

您必須至少定義一個Choice狀態中的規則。

Default (選用、建議)

如果未採用 Choices 中的任何轉換，則為狀態轉換至的名稱。

> **Important**
>
> Choice 狀態不支援 End 欄位。此外，這類狀態只會在其 Choices 欄位內使用 Next。

> **Tip**
>
> 若要部署使用Choice狀態的工作流程範例AWS 帳戶，請參閱模組 5-AWS Step Functions工作坊的選擇狀態和對應狀態。

選擇規則

Choice狀態必須有一個Choices字段，其值是一個非空數組。此陣列中的每個元素都是稱為 Choice Rule 的物件，其中包含下列項目：

- 比較 — 指定要比較之輸入變數的兩個欄位、比較類型以及要比較變數的值。選擇規則支援兩個變數之間的比較。在選擇規則中，透過附加Path至支援的比較運算子名稱，可以將變數的值與狀態輸入中的另一個值進行比較。比較中Variable和「路徑」欄位的值必須是有效的「參考路徑」。
- **Next**欄位 — 此欄位的值必須與狀態機器中的狀態名稱相符。

以下範例會檢查數值是否等於 1。

```
{
  "Variable": "$.foo",
  "NumericEquals": 1,
  "Next": "FirstMatchState"
}
```

以下範例會檢查字串是否等於 MyString。

```
{
  "Variable": "$.foo",
  "StringEquals": "MyString",
  "Next": "FirstMatchState"
}
```

以下範例會檢查字串是否大於 MyStringABC。

```
{
  "Variable": "$.foo",
  "StringGreaterThan": "MyStringABC",
  "Next": "FirstMatchState"
}
```

下列範例會檢查字串是否為 null。

```
{
 "Variable": "$.possiblyNullValue",
 "IsNull": true
}
```

下列範例顯示如何僅$.keyThatMightNotExist在因前面的IsPresent選擇StringEquals規則而存在時評估規則。

```
"And": [
 {
 "Variable": "$.keyThatMightNotExist",
 "IsPresent": true
 },
 {
 "Variable": "$.keyThatMightNotExist",
 "StringEquals": "foo"
```

```
 }
]
```

下列範例會檢查具有萬用字元的模式是否相符。

```
{
 "Variable": "$.foo",
 "StringMatches": "log-*.txt"
}
```

以下範例會檢查時間戳記是否等於 2001-01-01T12:00:00Z。

```
{
  "Variable": "$.foo",
  "TimestampEquals": "2001-01-01T12:00:00Z",
  "Next": "FirstMatchState"
}
```

下列範例會比較變數與 state 輸入中的另一個值。

```
{
 "Variable": "$.foo",
 "StringEqualsPath": "$.bar"
}
```

步驟函數會依Choices欄位中列出的順序檢查每個選擇規則。然後轉換為第一個「選擇規則」的 Next 欄位中指定的狀態，在該規則中變數會根據比較運算子來比對值。

支援下列比較運算子：

- And
- BooleanEquals,BooleanEqualsPath
- IsBoolean
- IsNull
- IsNumeric
- IsPresent
- IsString
- IsTimestamp

- Not
- NumericEquals,NumericEqualsPath
- NumericGreaterThan,NumericGreaterThanPath
- NumericGreaterThanEquals,NumericGreaterThanEqualsPath
- NumericLessThan,NumericLessThanPath
- NumericLessThanEquals,NumericLessThanEqualsPath
- Or
- StringEquals,StringEqualsPath
- StringGreaterThan,StringGreaterThanPath
- StringGreaterThanEquals,StringGreaterThanEqualsPath
- StringLessThan,StringLessThanPath
- StringLessThanEquals,StringLessThanEqualsPath
- StringMatches
- TimestampEquals,TimestampEqualsPath
- TimestampGreaterThan,TimestampGreaterThanPath
- TimestampGreaterThanEquals,TimestampGreaterThanEqualsPath
- TimestampLessThan,TimestampLessThanPath
- TimestampLessThanEquals,TimestampLessThanEqualsPath

對於這些運算子中的每一個，對應的值都必須是適當的類型：字串、數字、布林值或時間戳記。步驟函數不會嘗試將數值欄位與字串值相符。不過，由於時間戳記欄位邏輯上為字串，因此 StringEquals 比較子可以比對被視為時間戳記的欄位。

> **Note**
>
> 基於相互操作性，請勿假設數字比較適用於 IEEE 754-2008 binary64 資料類型所代表量級或精確度以外的值。尤其是，範圍 $[-2^{53}+1, 2^{53}-1]$ 之外的整數可能無法以預期的方式進行比較。
>
> 時間戳記 (例如 2016-08-18T17:33:00Z) 必須符合 RFC3339 設定檔 ISO 8601，以及進一步的限制：
>
> - 大寫字母 T 必須分開日期與時間部分。
> - 大寫字母 Z 必須表示不存在數字時間時區位移。

若要了解字串比較行為，請參閱 Java compareTo 文件。
And 和 Or 運算子的值必須是本身不得包含 Next 欄位的非空白選擇規則陣列。同樣地，Not 運算子的值必須是不得包含 Next 欄位的單一選擇規則。
您可以使用 And、Not 及 Or，建立複雜的巢狀選擇規則。不過，Next 欄位只能出現在最上層選擇規則中。
您可以使用比較運算子，對具有一或多個萬用字元 (「*」) 的樣式執行字串StringMatches比較。萬用字元會使用標準逸出\\ (Ex: “*”)。在比對期間，除了「*」之外，沒有任何特殊意義。

Choice 狀態範例

以下是 Choice 狀態的範例，以及它轉換至的其他狀態。

Note

您必須指定 $.type 欄位。如果狀態輸入不包含 $.type 欄位，則執行會失敗，且執行歷史記錄中會顯示一項錯誤。您只能在與常值相符的StringEquals欄位中指定字串。例如："StringEquals": "Buy"。

```
"ChoiceStateX": {
  "Type": "Choice",
  "Choices": [
    {
      "Not": {
        "Variable": "$.type",
        "StringEquals": "Private"
      },
      "Next": "Public"
    },
    {
      "Variable": "$.value",
      "NumericEquals": 0,
      "Next": "ValueIsZero"
    },
    {
      "And": [
        {
```

```
          "Variable": "$.value",
          "NumericGreaterThanEquals": 20
        },
        {
          "Variable": "$.value",
          "NumericLessThan": 30
        }
      ],
      "Next": "ValueInTwenties"
    }
  ],
  "Default": "DefaultState"
},

"Public": {
  "Type" : "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Foo",
  "Next": "NextState"
},

"ValueIsZero": {
  "Type" : "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Zero",
  "Next": "NextState"
},

"ValueInTwenties": {
  "Type" : "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Bar",
  "Next": "NextState"
},

"DefaultState": {
  "Type": "Fail",
  "Cause": "No Matches!"
}
```

在此範例中，狀態機器會從下列輸入值著手。

```
{
  "type": "Private",
  "value": 22
}
```

步驟函數轉換到ValueInTwenties狀態的基礎上，該value字段。

如果 Choice 狀態的 Choices 沒有相符項目，則會改為執行 Default 欄位中提供的狀態。如果未指定 Default 狀態，則執行會因錯誤而失敗。

等候

Wait 狀態 ("Type": "Wait") 會使狀態機器延遲，而無法繼續執行一段指定的時間。您可以選擇相對時間 (以狀態開始後的秒數指定)，或絕對結束時間 (以時間戳記形式指定)。

除了常見狀態欄位以外，Wait 狀態具有下列其中一個欄位。

Seconds

開始 Next 欄位中指定的狀態之前所要等待的時間 (以秒為單位)。您必須將時間指定為從 0 到 99999999 之間的正整數值。

Timestamp

開始 Next 欄位中指定的狀態之前，所要等待的絕對時間。

時間戳記必須符合 ISO 8601 的 RFC3339 設定檔，並且有以下的進一步限制：大寫 T 必須分隔日期和時間部分，以及大寫 Z 必須表示數字時區位移不存在，例如 2024-08-18T17:33:00Z。

> **Note**
>
> 目前，如果您將等待時間指定為時間戳記，Step Functions 會考慮時間值最多秒並截斷毫秒。

SecondsPath

開始 Next 欄位中指定的狀態之前所要等待的時間 (以秒為單)，而該欄位使用狀態輸入資料中的路徑來指定。

您必須為此欄位指定整數值。

TimestampPath

開始 Next 欄位中指定的狀態之前，所要等待的絕對時間，而該欄位使用狀態輸入資料中的路徑來指定。

> **Note**
>
> 您必須指定 Seconds、Timestamp、SecondsPath 或 TimestampPath 其中一項。此外，您可以為「標準工作流程」和「快速」工作流程指定的最長等待時間分別為一年五分鐘。

Wait 狀態範例

以下 Wait 狀態導致狀態機器延遲 10 秒。

```
"wait_ten_seconds": {
  "Type": "Wait",
  "Seconds": 10,
  "Next": "NextState"
}
```

在下一個範例中，Wait狀態會等到絕對時間：2024 年 3 月 14 日，世界標準時間上午 1:59。

```
"wait_until" : {
  "Type": "Wait",
  "Timestamp": "2024-03-14T01:59:00Z",
  "Next": "NextState"
}
```

您不需對此期間進行硬式編碼。舉例而言，假定是以下輸入資料：

```
{
  "expirydate": "2024-03-14T01:59:00Z"
}
```

使用參考路徑從輸入資料中選取 "expirydate" 值，即可從輸入中選取該值。

```
"wait_until" : {
    "Type": "Wait",
    "TimestampPath": "$.expirydate",
    "Next": "NextState"
}
```

Succeed

Succeed 狀態 ("Type": "Succeed") 會成功停止執行。對於除了停止執行以外，不會執行任何動作的 Choice 狀態分支而言，Succeed 狀態是很實用的目標。

因為 Succeed 狀態是終端狀態，所以沒有 Next 欄位，也不需要 End 欄位，如以下範例所示。

```
"SuccessState": {
  "Type": "Succeed"
}
```

Fail

一個Fail狀態 ("Type": "Fail") 停止狀態機的執行，並將其標記為失敗，除非它被Catch塊。

該Fail狀態只允許使用Type和Comment從一組字段一般狀態欄位。此外，Fail 狀態允許使用下列欄位。

Cause (選用)

描述錯誤原因的自訂字串。您可以為操作或診斷目的指定此欄位。

CausePath (選用)

如果您想從狀態輸入動態提供有關錯誤原因的詳細說明，請使用參考路徑，使用CausePath。解析後，參考路徑必須選取包含字串值的欄位。

您也可以指定CausePath使用內在函數返回一個字符串。這些內在函數是：狀態. 格式,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decode,狀態雜湊，以及States.UUID。

> **Important**
>
> - 您可以指定Cause或者CausePath，但不能同時處於「失敗」狀態定義中。
> - 基於資訊安全最佳實務，建議您從原因說明中移除任何敏感資訊或內部系統詳細資訊。

Error (選用)

您可以提供用於執行錯誤處理的錯誤名稱重試或者捕捉欄位。您也可以為操作或診斷目的提供錯誤名稱。

`ErrorPath` (選用)

如果您想從狀態輸入動態提供錯誤的名稱，請使用參考路徑，使用ErrorPath。解析後，參考路徑必須選取包含字串值的欄位。

您也可以指定ErrorPath使用內在函數返回一個字符串。這些內在函數是：狀態. 格式,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decode,狀態雜湊，以及States.UUID。

> **Important**
>
> - 您可以指定Error或者ErrorPath，但不能同時處於「失敗」狀態定義中。
> - 基於資訊安全最佳實務，建議您從錯誤名稱中移除任何敏感資訊或內部系統詳細資訊。

由於 Fail 狀態一律會結束狀態機器，因此沒有 Next 欄位，也不需要 End 欄位。

失敗狀態定義實務

下面的失敗狀態定義示例指定靜態Error和Cause字段值。

```
"FailState": {
  "Type": "Fail",
  "Cause": "Invalid response.",
  "Error": "ErrorA"
}
```

下列「失敗」狀態定義範例會動態使用參照路徑來解析Error和Cause字段值。

```
"FailState": {
  "Type": "Fail",
  "CausePath": "$.Cause",
  "ErrorPath": "$.Error"
}
```

下列「失敗」狀態定義範例使用狀態. 格式內在函數來指定Error和Cause動態字段值。

```
"FailState": {
  "Type": "Fail",
```

```
  "CausePath": "States.Format('This is a custom error message for {}, caused by {}.',
 $.Error, $.Cause)",
  "ErrorPath": "States.Format('{}', $.Error)"
}
```

平行

Parallelstate ("Type": "Parallel") 可用於在狀態機中添加單獨的執行分支。

除了常見狀態欄位以外，Parallel 狀態還引進下列額外欄位。

Branches (必要)

指定要以平行方式執行狀態機器的物件陣列。每個這類狀態機器物件必須具有名為 StartAt 和 States 的欄位，其意義完全如同狀態機器最上層中的欄位。

ResultPath (選用)

指定要將分支的輸出放置於何處 (在輸入中)。輸入會先依據 OutputPath 欄位 (如果有的話) 所指定篩選，而後做為狀態的輸出。如需詳細資訊，請參閱輸入和輸出處理。

ResultSelector (選用)

傳遞鍵值對的集合，其中值是靜態的或從結果中選擇的。如需詳細資訊，請參閱 ResultSelector。

Retry (選用)

稱為 Retrier 的物件陣列，這類物件可定義狀態發生執行時間錯誤時的重試政策。如需詳細資訊，請參閱 使用重試和使用 Catch 的狀態機器示例。

Catch (選用)

稱為 Catcher 的物件陣列，這類物件可定義在狀態發生執行時間錯誤以及其重試政策已耗盡或未定義時執行的備用狀態。如需詳細資訊，請參閱備用狀態。

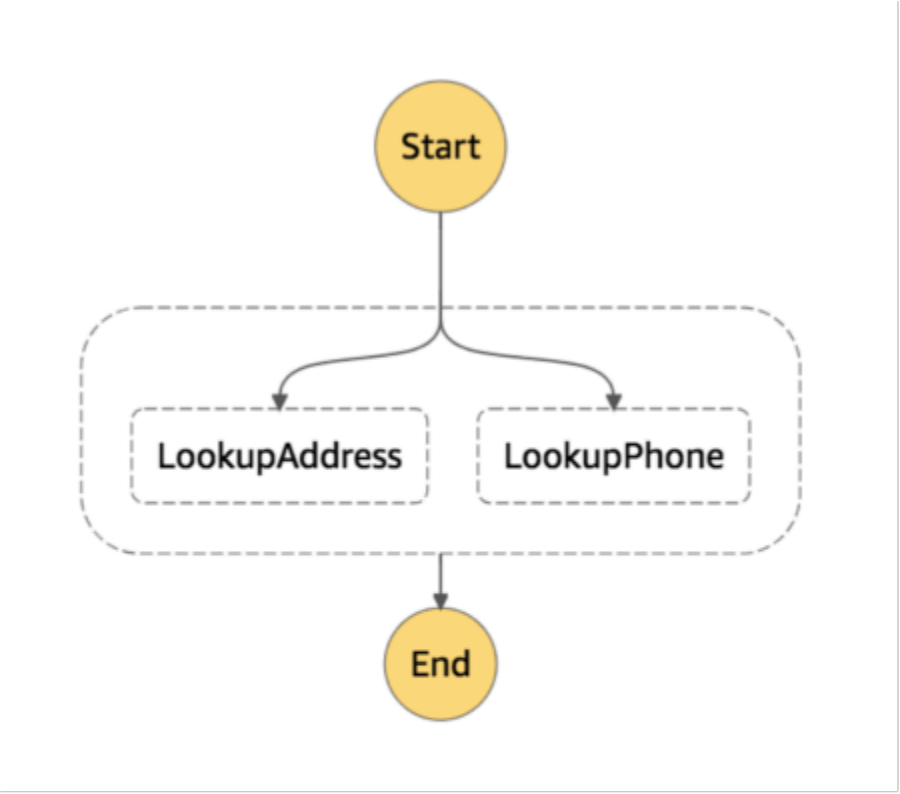
Parallel狀態會導 AWS Step Functions 致執行每個分支，從該分支StartAt字段中命名的狀態開始，盡可能同時執行，並等到所有分支終止（達到終端狀態），然後再處理該Parallel狀態的Next字段。

Parallel 狀態範例

```
{
```

```
  "Comment": "Parallel Example.",
  "StartAt": "LookupCustomerInfo",
  "States": {
    "LookupCustomerInfo": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
         "StartAt": "LookupAddress",
         "States": {
           "LookupAddress": {
             "Type": "Task",
             "Resource":
               "arn:aws:lambda:us-east-1:123456789012:function:AddressFinder",
             "End": true
           }
         }
        },
        {
          "StartAt": "LookupPhone",
          "States": {
            "LookupPhone": {
              "Type": "Task",
              "Resource":
                "arn:aws:lambda:us-east-1:123456789012:function:PhoneFinder",
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

在這個範例中，LookupAddress 和 LookupPhone 分支會以平行方式執行。以下是視覺化工作流程在 Step Functions 主控台中的外觀。



每個分支必須可獨立自主運作。一個 Parallel 狀態分支中的狀態不得有以該分支外部欄位為目標的 Next 欄位，而且分支外部的任何其他狀態也不能轉移到該分支。

Parallel 狀態輸入和輸出處理

Parallel 狀態會為每個分支提供一份自有的輸入資料 (可由 InputPath 欄位隨時修改)。它會產生陣列形式的輸出，其中包含分支輸出的每個分支都有一個元素。所有元素不需要都是相同的類型。按照一般方式使用 ResultPath 欄位，即可將輸出陣列插入輸入資料中 (以及整個當作 Parallel 狀態的輸出傳送) (請參閱輸入和輸出處理)。

```
{
  "Comment": "Parallel Example.",
  "StartAt": "FunWithMath",
```

```
  "States": {
    "FunWithMath": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Add",
          "States": {
            "Add": {
              "Type": "Task",
              "Resource": "arn:aws:states:us-east-1:123456789012:activity:Add",
              "End": true
            }
          }
        },
        {
          "StartAt": "Subtract",
          "States": {
            "Subtract": {
              "Type": "Task",
              "Resource": "arn:aws:states:us-east-1:123456789012:activity:Subtract",
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

如果 FunWithMath 狀態取得陣列 [3, 2] 做為輸入，則 Add 和 Subtract 狀態都會收到該陣列做為輸入。Add和 Subtract task 的輸出將是數組元素 3 和 2 之間的總和和差異1，5而Parallel狀態的輸出將是一個數組。

```
[ 5, 1 ]
```

> **Tip**
>
> 如果您在狀態機器中使用的「平行」或「對映」狀態傳回陣列陣列，您可以將它們轉換為具有ResultSelector欄位的平面陣列。如需詳細資訊，請參閱 扁平化陣列的陣列。

錯誤處理

如有任何分支因為未處理的錯誤或經由轉移到 `Fail` 狀態而失敗，則整個 `Parallel` 狀態會被視為失敗，而且其所有分支都會停止。如果錯誤不是由`Parallel`狀態本身處理，Step Functions 會停止執行並出現錯誤。

> **Note**
>
> 當 parallel 狀態失敗時，叫用的 Lambda 函數會繼續執行，且不會停止處理工作權杖的活動工作者。
>
> - 若要停止長時間執行的活動，請使用活動訊號來偵測其分支是否已由 Step Functions 停止，並停止正在處理工作的 Worker。如果狀態失敗，則呼叫 `SendTaskHeartbeat`、`SendTaskSuccess` 或 `SendTaskFailure` 會拋出錯誤。請參閱活動訊號錯誤。
> - 執行 Lambda 函數無法停止。如果您已實作後援，請使用`Wait`狀態，以便在 Lambda 函數完成後進行清理工作。

Map

使用狀`Map`態為資料集中的每個項目執行一組工作流程步驟。`Map`狀態的反覆項目會 parallel 執行，因此可以快速處理資料集。 `Map`狀態可以使用各種輸入類型，包括 JSON 陣列、Amazon S3 物件清單或 CSV 檔案。

Step Functions 提供兩種類型的處理模式，可在工作流程中使用`Map`狀態：內嵌模式和分散式模式。

如需這些模式以及如何在任一模式中使用`Map`狀態的相關資訊，請參閱下列主題：

- 對映狀態處理模式
  - 在內嵌模式中使用對應狀態
  - 在分散式模式中使用對應狀態

> **Tip**
>
> 若要部署使用`Map`狀態的工作流程範例AWS 帳戶，請參閱模組 5-AWS Step Functions工作坊的選擇狀態和對應狀態。

對映狀態處理模式

Step Functions 會根據您要如何處理資料集中的項目，提供Map狀態的下列處理模式。

- 內聯-有限並發模式。在此模式下，狀態的每個版序都會在包含Map狀態的工作流程前後關聯中Map執行。Step Functions 會將這些反覆項目的執行歷程記錄新增至父工作流程的執行歷程記錄。依預設，Map狀態會以內嵌模式執行。

  在這種模式下，Map狀態只接受 JSON 數組作為輸入。此外，此模式最多支持 40 個並發迭代。

  如需詳細資訊，請參閱在內嵌模式中使用對應狀態。
- 分佈式-高並發模式。在此模式下，Map狀態會將每個反覆作為子工作流程執行執行執行執行來執行，最多可達 10,000 個 parallel 子工作流程執行的高並行性。每個子工作流程執行都有自己的執行記錄，與父工作流程的執行歷程記錄不同。

  在此模式下，Map狀態可以接受 JSON 陣列或 Amazon S3 資料來源 (例如 CSV 檔案) 作為其輸入。

  如需詳細資訊，請參閱在分散式模式中使用對應狀態。

您應該使用的模式取決於您要如何處理資料集中的項目。如果工作流程的執行歷程記錄不超過 25,000 個項目，或者您不需要超過 40 個並行迭代，請在內嵌模式中使用Map狀態。

當您需要協調符合下列任何條件組合的大規模 parallel 工作負載時，請使用分散式模式中的Map狀態：

- 資料集的大小超過 256 KB。
- 工作流程的執行事件歷程記錄超過 25,000 個項目。
- 您需要超過 40 個 parallel 迭代的並發性。

主題

內聯模式和分佈式模式差異

下表反白「內嵌」和「分散式」模式之間的差異。

內嵌模式	分散式模式
Supported data sources	
接受從工作流程中上一個步驟傳遞的 JSON 陣列作為輸入。	接受下列資料來源作為輸入： • 從工作流程中的上一個步驟傳遞的 JSON 陣列 • 包含數組的 Amazon S3 存儲桶中的 JSON 文件 • Amazon S3 存儲桶中的 CSV 文件 • Amazon S3 對象列表 • Amazon S3 清查
Map iterations	
在此模式下，狀態的每個版序都會在包含 Map狀態的工作流程前後關聯中Map執行。Step Functions 會將這些反覆項目的執行歷程記錄新增至父工作流程的執行歷程記錄。	在此模式下，Map狀態會將每個反覆作為子工作流程執行執行執行執行來執行，最多可達 10,000 個 parallel 子工作流程執行的高並行性。每個子工作流程執行都有自己的執行記錄，與父工作流程的執行歷程記錄不同。
Maximum concurrency for parallel iterations	
可讓您盡可能同時執行多達 40 個反覆項目。	可讓您執行多達 10,000 個 parallel 子工作流程執行，一次處理數百萬個資料項目。
Input payload and event history sizes	
對輸入有效負載大小強制執行 256 KB 的限制，並在執行事件歷史記錄中強制執行 25,000 個項目。	可讓您克服承載大小限制，因為該Map狀態可以直接從 Amazon S3 資料來源讀取輸入。 在此模式下，您也可以克服執行歷程記錄的限制，因為由Map狀態開始的子工作流程執行會維護它們自己的執行歷程記錄與父工作流程的執行歷程記錄分開。
Monitoring and observability	

內嵌模式

您可以從主控台或叫用 GetExecutionHistory API 動作來檢閱工作流程的執行歷程記錄。

您還可以通過 CloudWatch 和 X-Ray 查看執行歷史記錄。

分散式模式

當您在分散式模式下執行Map狀態時，「Step Functions」會建立「對應執行」資源。Map Run 是指分散式地圖狀態啟動的一組子工作流程執行。您可以在「Step Functions」主控台中檢視地圖執行。您也可以叫用 DescribeMapRun API 動作。「地圖執行」也會向其發出CloudWatch度量。

如需詳細資訊，請參閱 檢查分散式地圖狀態執行的對應執行。

在內嵌模式中使用對應狀態

依預設，Map狀態會以內嵌模式執行。在內聯模式下，地圖狀態只接受 JSON 數組作為輸入。它會從工作流程中的上一個步驟接收此陣列。在此模式下，狀態的每個版序都會在包含Map狀態的工作流程前後關聯中Map執行。Step Functions 會將這些反覆項目的執行歷程記錄新增至父工作流程的執行歷程記錄。

在這種模式下，Map狀態最多支持 40 個並發迭代。

設置為內聯的Map狀態被稱為內聯地圖狀態。如果工作流程的執行歷程記錄不超過 25,000 個項目，或者您不需要超過 40 個並行迭代，請在內嵌模式中使用Map狀態。

如需使用內嵌對應狀態的簡介，請參閱自學課程使用內嵌對應狀態重複動作。

目錄

本主題的關鍵概念

內嵌模式

狀態的有限並發模式。Map在此模式下，狀態的每個版序都會在包含Map狀態的工作流程前後關聯中Map執行。Step Functions 會將這些反覆項目的執行歷程記錄新增至父工作流程的執行歷程記錄。 Map狀態默認情況下以內聯模式運行。

此模式僅接受 JSON 數組作為輸入，並支持多達 40 個並發迭代。

內聯映射狀態

設定為「內嵌」模式的Map狀態。

地圖工作流

Map狀態會針對每個版序執行的一組步驟。

映射狀態迭代

在Map狀態內部定義的工作流程的重複。

內嵌對應狀態欄位

若要在工作流程中使用內嵌對應狀態，請指定一或多個這些欄位。除了一般狀態欄位之外，您還可以指定這些欄位。

`Type` (必要)

設定狀態的類型，例如Map。

`ItemProcessor` (必要)

包含下列指定Map狀態處理模式和定義的 JSON 物件。

定義包含處理每個陣列項目時要重複的一組步驟。

- `ProcessorConfig`— 選用的 JSON 物件，指定Map狀態的處理模式。此物件包含Mode子欄位。此欄位預設為INLINE，此欄位使用內嵌模式下的Map狀態。

  在這種模式下，任何迭代的失敗都會導致Map狀態失敗。當Map狀態失敗時，所有版序都會停止。
- `StartAt`— 指定指示工作流程中第一個狀態的字串。此字串區分大小寫，且必須與其中一個狀態物件的名稱相符。此狀態會先針對資料集中的每個項目執行。您提供給Map狀態的任何執行輸入會先傳遞至StartAt狀態。

- `States`— 包含逗號分隔狀態集的 JSON 物件。在此物件中，您可以定義Map workflow.

  > **Note**
  >
  > - `ItemProcessor`欄位中的狀態只能相互轉換。`ItemProcessor`欄位外的任何狀態都無法轉換為其中的狀態。
  > - 此`ItemProcessor`欄位會取代現在已取代的`Iterator`欄位。雖然您可以繼續包含使用`Iterator`欄位的`Map`狀態，但我們強烈建議您將此欄位取代為`ItemProcessor`。
  >
  >   Step Functions 本地目前不支持該`ItemProcessor`字段。我們建議您使用`Iterator`欄位與 Step Functions 本機。

`ItemsPath` (選用)

使用JsonPath語法指定參考路徑。此路徑選擇包含狀態輸入內的項目數組的 JSON 節點。如需詳細資訊，請參閱ItemsPath。

`ItemSelector` (選用)

覆寫輸入陣列項目的值，然後再傳遞給每個`Map`狀態迭代。

在此欄位中，您可以指定包含索引鍵值組集合的有效 JSON。這些配對可以包含下列任何項目：

- 您在狀態機定義中定義的靜態值。
- 使用路徑從狀態輸入中選取的值。
- 從上下文對象訪問的值。

如需詳細資訊，請參閱ItemSelector。

此`ItemSelector`欄位會取代現在已取代的`Parameters`欄位。雖然您可以繼續包含使用`Parameters`欄位的`Map`狀態，但我們強烈建議您將此欄位取代為`ItemSelector`。

`MaxConcurrency` (選用)

指定整數值，提供可 parallel 執行之`Map`狀態反覆次數的上限。例如，`MaxConcurrency`值 10 將`Map`狀態限制為一次執行 10 個並行迭代。

> **Note**
>
> 並發迭代可能會受到限制。發生這種情況時，在先前的迭代完成之前，某些迭代才會開始。當您的輸入陣列有超過 40 個項目時，發生這種情況的可能性就會增加。
> 為了實現更高的並發性，請考慮在分散式模式中使用對應狀態。

預設值為0，此值不會限制連續性。Step Functions 盡可能同時調用迭代。

的MaxConcurrency值會針對每個陣列元素1叫用ItemProcessor一次。陣列中的項目會依照其在輸入中出現的順序進行處理。Step Functions 不會啟動一個新的迭代，直到它完成前一次迭代。

MaxConcurrencyPath (選用)

如果您想要使用MaxConcurrencyPath參考路徑從狀態輸入動態提供最大並行值，請使用。解析後，參考路徑必須選取值為非負整數的欄位。

> **Note**
>
> 狀Map態不能同時包含MaxConcurrency和MaxConcurrencyPath。

ResultPath (選用)

指定在輸入中存儲狀Map態迭代輸出的位置。然後地圖狀態過濾輸入由OutputPath字段指定，如果指定。然後，它使用過濾的輸入作為狀態的輸出。如需詳細資訊，請參閱輸入和輸出處理。

ResultSelector (選用)

傳遞鍵值對的集合，其中的值是靜態的或從結果中選擇的。如需詳細資訊，請參閱ResultSelector。

> **Tip**
>
> 如果您在狀態機器中使用的「平行」或「對映」狀態傳回陣列陣列，您可以將它們轉換為具有ResultSelector欄位的平面陣列。如需詳細資訊，請參閱扁平化陣列的陣列。

Retry (選用)

物件陣列 (稱為「擷取器」)，可定義重試原則。當狀態遇到執行階段錯誤時，會使用重試原則。如需詳細資訊，請參閱使用重試和使用 Catch 的狀態機器示例。

> **Note**
>
> 如果您為「內嵌對應」狀態定義「擷取器」，則重試原則會套用至所有Map狀態版序，而不是僅套用失敗的版序。例如，您的Map狀態包含兩個成功的版序和一個失敗的版序。如果您已定義Map狀態的Retry欄位，則重試原則會套用至所有三個Map狀態反覆項目，而不是僅套用到失敗的反覆項目。

`Catch` (選用)

稱為 Catcher 的物件陣列，可定義後援狀態。如果狀態遇到執行階段錯誤且沒有重試原則，或者其重試原則已用盡，則會執行捕捉器。如需詳細資訊，請參閱備用狀態。

已取代欄位

> **Note**
>
> 雖然您可以繼續包含使用下列欄位的Map狀態，但我們強烈建議您取代Iterator為ItemProcessorParameters和ItemSelector。

`Iterator`

指定 JSON 物件，此物件會定義處理陣列中每個元素的一組步驟。

`Parameters`

指定索引鍵值配對的集合，其中值可以包含下列任何項目：

- 您在狀態機定義中定義的靜態值。
- 使用路徑從輸入中選取的值。

內聯映射狀態示例

請考慮以內嵌模式執行Map狀態的下列輸入資料。

```
{
  "ship-date": "2016-03-14T01:59:00Z",
  "detail": {
    "delivery-partner": "UQS",
    "shipped": [
      { "prod": "R31", "dest-code": 9511, "quantity": 1344 },
      { "prod": "S39", "dest-code": 9511, "quantity": 40 },
      { "prod": "R31", "dest-code": 9833, "quantity": 12 },
      { "prod": "R40", "dest-code": 9860, "quantity": 887 },
      { "prod": "R40", "dest-code": 9511, "quantity": 1220 }
    ]
  }
```

```
}
```

根據先前的輸入，下列範例中的 Map state 會針對shipped欄位中陣列的每個項目叫用名為ship-val一次的AWS Lambda函數。

```
"Validate All": {
    "Type": "Map",
    "InputPath": "$.detail",
    "ItemProcessor": {
        "ProcessorConfig": {
            "Mode": "INLINE"
        },
        "StartAt": "Validate",
        "States": {
            "Validate": {
                "Type": "Task",
                "Resource": "arn:aws:states:::lambda:invoke",
                "OutputPath": "$.Payload",
                "Parameters": {
                    "FunctionName": "arn:aws:lambda:us-
east-2:123456789012:function:ship-val:$LATEST"
                },
                "End": true
            }
        }
    },
    "End": true,
    "ResultPath": "$.detail.shipped",
    "ItemsPath": "$.shipped"
}
```

Map狀態的每個反覆運算都會傳送陣列中的項目，並與ItemsPath欄位一起選取，做為 ship-val Lambda 函數的輸入。下列值是Map狀態傳送至 Lambda 函數叫用之輸入的範例：

```
{
  "prod": "R31",
  "dest-code": 9511,
  "quantity": 1344
}
```

完成時，Map 狀態的輸出為 JSON 陣列，其中每個項目都是反覆運算的輸出。在這種情況下，此陣列包含 ship-val Lambda 函數的輸出。

內聯映射狀態示例 **ItemSelector**

假設上一個範例中的 ship-val Lambda 函數也需要有關貨件快遞的資訊。此資訊是陣列中每次反覆運算的項目之外的資訊。您可以包括來自輸入的資訊，以及Map狀態目前版序的特定資訊。請注意下列範例中的ItemSelector欄位：

```
"Validate-All": {
  "Type": "Map",
  "InputPath": "$.detail",
  "ItemsPath": "$.shipped",
  "MaxConcurrency": 0,
  "ResultPath": "$.detail.shipped",
  "ItemSelector": {
    "parcel.$": "$$.Map.Item.Value",
    "courier.$": "$.delivery-partner"
  },
  "ItemProcessor": {
    "StartAt": "Validate",
    "States": {
      "Validate": {
        "Type": "Task",
 "Resource": "arn:aws:lambda:us-east-1:123456789012:function:ship-val",
        "End": true
      }
    }
  },
  "End": true
}
```

該ItemSelector塊將輸入替換為 JSON 節點的迭代。此節點包含來自上下文對象的當前項目數據和來自Map狀態輸入delivery-partner字段的快遞信息。以下是輸入到單個迭代的示例。狀Map態會將此輸入傳遞至 ship-val Lambda 函數的叫用。

```
{
  "parcel": {
    "prod": "R31",
    "dest-code": 9511,
    "quantity": 1344
   },
   "courier": "UQS"
}
```

在上一個內嵌對應狀態範例中，ResultPath欄位會以與輸入相同的格式產生輸出。但是，它會使用陣列覆寫detail.shipped欄位，其中每個元素都是每次迭代 ship-val Lambda 叫用的輸出。

如需有關使用「內嵌對應」狀態及其欄位的詳細資訊，請參閱下列內容。

- 使用內嵌對應狀態重複動作
- Step Functions 中的輸入和輸出處理
- ItemsPath
- Map 狀態的內容物件資料

內聯Map狀態輸入和輸出處理

對於給定的Map狀態，InputPath選取狀態輸入的子集。

Map狀態的輸入必須包含 JSON 陣列。狀Map態會針對陣列中的每個項目執行一次ItemProcessor區段。如果您指定ItemsPath欄位，Map狀態會選取輸入中的哪個位置來尋找要迭代的陣列。如果未指定，則 ItemsPath 的值為 $，而 ItemProcessor 區段預期陣列是唯一的輸入。如果您指定ItemsPath欄位，則其值必須是參考路徑。狀Map態將此路徑套用至有效輸入之後InputPath。必ItemsPath須識別其值為 JSON 陣列的欄位。

每次迭代的輸入，默認情況下，是由該ItemsPath值標識的數組字段的單個元素。您可以使用ItemSelector欄位覆寫此值。

完成時，Map 狀態的輸出為 JSON 陣列，其中每個項目都是反覆運算的輸出。

如需有關內嵌對應狀態輸入和輸出的詳細資訊，請參閱下列內容：

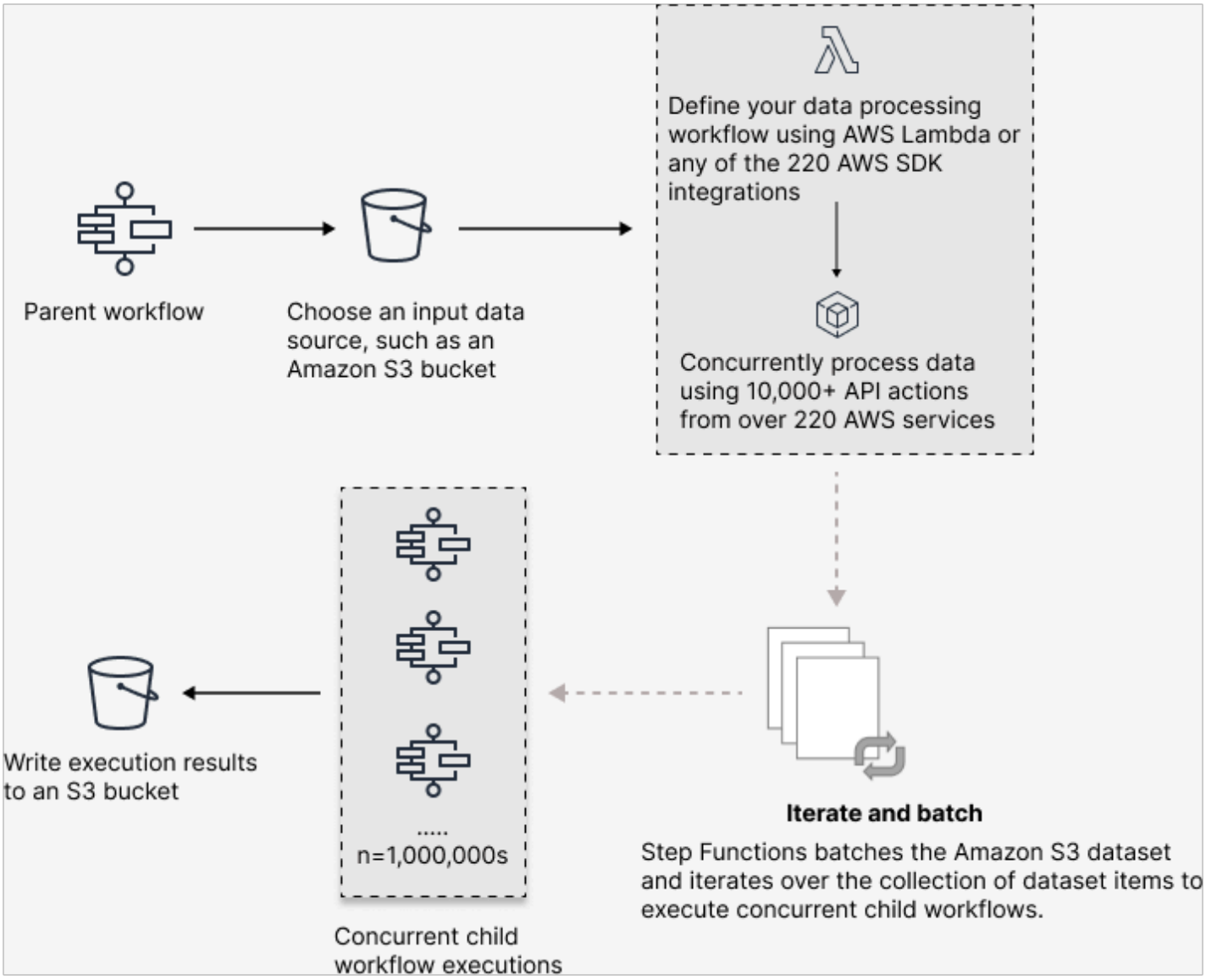
- 使用內嵌對應狀態重複動作
- 內聯映射狀態示例 ItemSelector
- Step Functions 中的輸入和輸出處理
- Map 狀態的內容物件資料
- 使用「地圖」狀態動態處理資料

在分散式模式中使用 Map 狀態來協調大規模的 parallel 工作負載

使用 Step Functions，您可以協調大規模的 parallel 工作負載以執行工作，例如按需處理半結構化資料。這些 parallel 工作負載可讓您同時處理存放在 Amazon S3 中的大規模資料來源。例如，您可以處理包含大量資料的單一 JSON 或 CSV 檔案。或者，您可能會處理大量的 Amazon S3 物件集。

若要在工作流程中設定大規模的 parallel 工作負載，請在「分散式」模式中包含Map狀態。Map 狀態會同時處理資料集中的項目。設置為分佈式的Map狀態稱為分佈式地圖狀態。在分散式模式下，狀Map態允許高並行處理。在「分散式」模式中，Map狀態會以稱為子工作流程執行的版序處理資料集中的項目。您可以指定可以 parallel 執行的子工作流程執行數目。每個子工作流程執行都有自己的執行記錄，與父工作流程的執行歷程記錄不同。如果未指定，「Step Functions」會 parallel 執行 10,000 個 parallel 子工作流程執行。

下圖說明如何在工作流程中設定大規模的 parallel 工作負載。



在本主題中

- 重要用語
- 分散式對應狀態定義範例

- 執行分散式地圖的權限
- 分散式地圖狀態欄位
- 後續步驟

重要用語

分散式模式

「對應」狀態的處理模式。在此模式下，Map狀態的每個版序都會以啟用高並行性的子工作流程執行方式執行來執行。每個子工作流程執行都有自己的執行歷程記錄，這與父工作流程的執行歷程記錄不同。此模式支援從大規模 Amazon S3 資料來源讀取輸入。

分散式地圖狀態

設定為分散式處理模式的對應狀態。

地圖工作流

Map狀態執行的一組步驟。

父工作流程

包含一或多個分散式地圖狀態的工作流程。

子工作流程執行

「分散式貼圖」狀態的版序。子工作流程執行具有自己的執行歷程記錄，與父工作流程的執行歷程記錄不同。

地圖運行

當您在分散式模式下執行Map狀態時，「Step Functions」會建立「對應執行」資源。Map Run 是指分散式地圖狀態啟動的一組子工作流程執行，以及控制這些執行的執行階段設定。Step Functions 將 Amazon 資源名稱（ARN）分配給您的地圖運行。您可以在 Step Functions 控制台中檢查地圖運行。您也可以叫用 `DescribeMapRun` API 動作。「地圖執行」也會向其發出CloudWatch度量。

如需詳細資訊，請參閱 檢查地圖運行。

分散式對應狀態定義範例

當您需要協調符合下列任何條件組合的大規模 parallel 工作負載時，請使用分散式模式中的Map狀態：

- 資料集的大小超過 256 KB。
- 工作流程的執行事件歷程記錄超過 25,000 個項目。
- 您需要超過 40 個 parallel 迭代的並發性。

下列分散式地圖狀態定義範例會將資料集指定為存放在 Amazon S3 儲存貯體中的 CSV 檔案。它也會指定 Lambda 函數來處理 CSV 檔案每一列中的資料。由於此範例使用 CSV 檔案，因此也會指定 CSV 欄標題的位置。若要檢視此範例的完整狀態機定義，請參閱使用分散式地圖複製大規模 CSV 資料教學課程。

```
{
  "Map": {
    "Type": "Map",
    "ItemReader": {
      "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
      },
      "Resource": "arn:aws:states:::s3:getObject",
      "Parameters": {
        "Bucket": "Database",
        "Key": "csv-dataset/ratings.csv"
      }
    },
    "ItemProcessor": {
      "ProcessorConfig": {
        "Mode": "DISTRIBUTED",
        "ExecutionType": "EXPRESS"
      },
      "StartAt": "LambdaTask",
      "States": {
        "LambdaTask": {
          "Type": "Task",
          "Resource": "arn:aws:states:::lambda:invoke",
          "OutputPath": "$.Payload",
          "Parameters": {
            "Payload.$": "$",
            "FunctionName": "arn:aws:lambda:us-
east-2:123456789012:function:processCSVData"
          },
          "End": true
        }
```

```
      }
    },
    "Label": "Map",
    "End": true,
    "ResultWriter": {
      "Resource": "arn:aws:states:::s3:putObject",
      "Parameters": {
        "Bucket": "myOutputBucket",
        "Prefix": "csvProcessJobs"
      }
    }
  }
}
```

執行分散式地圖的權限

當您在工作流程中包含分散式對應狀態時，Step Functions 需要適當的權限，才能允許狀態機器角色呼叫分散式對應狀態的 StartExecution API 動作。

以下 IAM 政策示例授予狀態機器角色執行分散式地圖狀態所需的最少權限。

> **Note**
>
> 請確定您以使*stateMachineName*用分散式地圖狀態的狀態機器名稱取代。例如 arn:aws:states:*us-east-2*:*123456789012*:stateMachine:*mystateMachine*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:stateMachine:stateMachineName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:region:accountID:execution:stateMachineName:*"
    }
  ]
}
```

此外，您需要確保您擁有存取分散式地圖狀態中使用的 AWS 資源 (例如 Amazon S3 儲存貯體) 所需的最低權限。如需相關資訊，請參閱使用分散式地圖狀態的 IAM 政策。

分散式地圖狀態欄位

若要在工作流程中使用「分散式地圖」狀態，請指定其中一個或多個欄位。除了一般狀態欄位之外，您還可以指定這些欄位。

`Type` (必要)

設定狀態的類型，例如`Map`。

`ItemProcessor` (必要)

包含下列指定`Map`狀態處理模式和定義的 JSON 物件。

- `ProcessorConfig`— 指定狀態組`Map`態的 JSON 物件。此物件包含下列子欄位：
  - `Mode`— 設定為使**`DISTRIBUTED`**用分散式模式中的`Map`狀態。

    > **Note**
    >
    > 目前，如果您使用 Express 工作流程內的`Map`狀態，則無法將設定`Mode`為`DISTRIBUTED`。但是，如果您在「標準」工作流程中使用`Map`狀態，則可以`Mode`將「」設定為`DISTRIBUTED`。

  - `ExecutionType`— 將「對映」工作流程的執行類型指定為「標準」或「快速」。如果您`DISTRIBUTED`為`Mode`子欄位指定了此欄位，則必須提供此欄位。如需工作流程類型的詳細資訊，請參閱標準與快速工作流程。
- `StartAt`— 指定指示工作流程中第一個狀態的字串。此字串區分大小寫，且必須與其中一個狀態物件的名稱相符。此狀態會先針對資料集中的每個項目執行。您提供給`Map`狀態的任何執行輸入會先傳遞至`StartAt`狀態。
- `States`— 包含逗號分隔狀態集的 JSON 物件。在此物件中，您可以定義Map workflow.

`ItemReader`

指定資料集及其位置。狀Map態會從指定的資料集接收其輸入資料。

在分散式模式中，您可以使用從先前狀態傳遞的 JSON 承載，或使用大型 Amazon S3 資料來源做為資料集。如需詳細資訊，請參閱 ItemReader。

`ItemsPath` (選用)

使用JsonPath語法來指定參考路徑，該 JSON 節點包含狀態輸入內的項目陣列。

在分散式模式中，只有當您使用上一個步驟中的 JSON 陣列做為狀態輸入時，才能指定此欄位。如需詳細資訊，請參閱 ItemsPath。

`ItemSelector` (選用)

覆寫個別資料集項目的值，然後再傳遞至每個Map狀態反覆運算。

在此欄位中，您可以指定包含索引鍵值組集合的有效 JSON 輸入。這些配對可以是您在狀態機器定義中定義的靜態值、使用路徑從狀態輸入中選取的值，或是從前後關聯物件存取的值。如需詳細資訊，請參閱 ItemSelector。

`ItemBatcher` (選用)

指定以批次方式處理資料集項目。然後，每個子工作流程執行都會接收這些項目的批次作為輸入。如需詳細資訊，請參閱 ItemBatcher。

`MaxConcurrency` (選用)

指定可以 parallel 執行的子工作流程執行數目。解譯器最多只允許指定數目的 parallel 子工作流程執行。如果您未指定並行值或將其設定為零，則 Step Functions 不會限制並行執行 10,000 個 parallel 子工作流程執行。

> **Note**
>
> 雖然您可以為 parallel 子工作流程執行指定較高的並行限制，但我們建議您不要超過下游 AWS 服務的容量，例如。 AWS Lambda

`MaxConcurrencyPath` (選用)

如果您想要使用MaxConcurrencyPath參考路徑從狀態輸入動態提供最大並行值，請使用。解析後，參考路徑必須選取值為非負整數的欄位。

> **Note**
>
> 狀Map態不能同時包含MaxConcurrency和MaxConcurrencyPath。

ToleratedFailurePercentage (選用)

定義在地圖執行中容許的失敗項目百分比。如果地圖執行超過此百分比，則會自動失敗。Step Functions 會計算失敗項目的百分比，因為失敗或逾時項目總數除以項目總數的結果。您必須指定一個介於零和 100 之間的值。如需詳細資訊，請參閱 分散式對應狀態的容許失敗臨界值。

ToleratedFailurePercentagePath (選用)

如果您想要使用ToleratedFailurePercentagePath參照路徑，從狀態輸入動態提供容錯失敗百分比值，請使用。解析後，參考路徑必須選取值介於 0 和 100 之間的欄位。

ToleratedFailureCount (選用)

定義地圖執行中要容許的失敗項目數。如果地圖執行超過此數目，則會自動失敗。如需詳細資訊，請參閱 分散式對應狀態的容許失敗臨界值。

ToleratedFailureCountPath (選用)

如果您想要使用ToleratedFailureCountPath參照路徑，從狀態輸入動態提供容忍失敗計數值，請使用。解析後，參考路徑必須選取值為非負整數的欄位。

Label (選用)

唯一識別Map狀態的字串。對於每個地圖運行，Step Functions 將標籤添加到地圖運行 ARN。以下是具demoLabel有名為的自訂標籤的 Map Run ARN 的範例：

```
arn:aws:states:us-east-1:123456789012:mapRun:demoWorkflow/
demoLabel:3c39a231-69bb-3d89-8607-9e124eddbb0b
```

如果您未指定標籤，Step Functions 會自動產生唯一的標籤。

> **Note**
>
> 標籤長度不能超過 40 個字元，在狀態機器定義中必須是唯一的，且不能包含下列任何字元：
>
> - 空白字元
> - 萬用字元 (? *)

- 括號字元 (< > { } [ ])
- 特殊字元 (: ; , \ | ^ ~ $ # % & ` ")
- 控制字符 (\\u0000-\\u001f 或 \\u007f-\\u009f)。

Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

`ResultWriter` (選用)

指定 Step Functions 寫入所有子工作流程執行結果的 Amazon S3 位置。

Step Functions 會合併所有子工作流程執行資料，例如執行輸入與輸出、ARN 及執行狀態。然後，它會將具有相同狀態的執行匯出到指定 Amazon S3 位置的個別檔案。如需詳細資訊，請參閱 ResultWriter。

如果您不匯出Map狀態結果，它會傳回所有子工作流程執行結果的陣列。例如：

```
[1, 2, 3, 4, 5]
```

`ResultPath` (選用)

指定在輸入中放置迭代輸出的位置。然後，在輸入作為狀態的輸出傳遞之前，按照`OutputPath`字段指定（如果存在）進行過濾。如需詳細資訊，請參閱輸入和輸出處理。

`ResultSelector` (選用)

傳遞鍵值對的集合，其中值是靜態的或從結果中選擇的。如需詳細資訊，請參閱 ResultSelector。

> **Tip**
>
> 如果您在狀態機器中使用的「平行」或「對映」狀態傳回陣列陣列，您可以將它們轉換為具有ResultSelector欄位的平面陣列。如需詳細資訊，請參閱 扁平化陣列的陣列。

`Retry` (選用)

物件陣列 (稱為「擷取器」)，可定義重試原則。如果狀態遇到執行階段錯誤，執行會使用重試原則。如需詳細資訊，請參閱 使用重試和使用 Catch 的狀態機器示例。

> **Note**
>
> 如果您定義 [分散式對應] 狀態的 [擷取器]，則重試原則會套用至所有已啟動Map狀態的子工作流程執行。例如，假設您的Map狀態開始了三個子工作流程執行，其中一個執行失敗。發生失敗時，執行會使用Retry欄位 (如果已定義) 做為Map狀態。重試原則適用於所有子工作流程執行，而不僅僅是失敗的執行。如果一個或多個子工作流程執行失敗，則 Map Run 會失敗。
>
> 當您重試某個Map狀態時，它會建立新的 Map Run。

`Catch` (選用)

稱為 Catcher 的物件陣列，可定義後援狀態。Step Functions 使用中定義的捕獲器，Catch如果狀態遇到運行時錯誤。發生錯誤時，執行會先使用中Retry定義的任何擷取器。如果重試原則未定義或已用盡，則執行會使用其 Catcher (如果已定義)。如需詳細資訊，請參閱備用狀態。

後續步驟

若要繼續學習有關分散式地圖狀態的更多資訊，請參閱下列資源：

- 輸入和輸出處理

  若要設定分散式對應狀態接收的輸入及其產生的輸出，Step Functions 提供下列欄位：

  - ItemReader
  - ItemsPath
  - ItemSelector
  - ItemBatcher
  - ResultWriter
  - 剖析輸入 CSV 檔案

  除了這些欄位之外，「Step Functions」還可讓您定義「分散式對應」的容許失敗臨界值。此值可讓您指定失敗項目的最大數目或百分比做為 Map Run 的失敗臨界值。如需設定容許失敗臨界值的詳細資訊，請參閱分散式對應狀態的容許失敗臨界值。
- 使用分散式地圖狀態

  請參閱下列教學課程和範例專案，以開始使用分散式地圖狀態。

  - 開始使用分散式地圖狀態

- 使用 Lambda 函數處理整批資料
- 使用 Lambda 函數處理個別資料項目
- 範例專案：使用分散式地圖處理 CSV 檔案
- 範例專案：使用分散式地圖處理 Amazon S3 儲存貯體中的資料

- 檢查分散式對應狀態執行

  「Step Functions」主控台提供「對應執行詳細資訊」頁面，其中顯示與「分散式對應」狀態執行相關的所有資訊。如需有關如何檢查此頁面上顯示之資訊的資訊，請參閱檢查地圖運行。

分散式對應狀態的容許失敗臨界值

當您協調大規模的 parallel 工作負載時，您也可以定義容忍的故障閾值。此值可讓您指定失敗項目的最大數目或百分比做為 Map Run 的失敗臨界值。根據您指定的值，如果 Map Run 超過臨界值，則會自動失敗。如果您同時指定這兩個值，則工作流程會在超過任一值時失敗。

指定臨界值可協助您在整個 Map Run 失敗之前失敗特定數目的項目。當「對映執行」因為超過指定的臨界值而失敗時，「Step Functions」會傳回`States.ExceedToleratedFailureThreshold`錯誤。

> **Note**
>
> 即使在超過容許的失敗閾值之後，但在 Map Run 失敗之前，Step Functions 仍可繼續在 Map Run 中執行子工作流程。

若要在 Workflow Studio 中指定臨界值，請在 [執行時期設定] 欄位下的 [其他組態] 中選取 [設定容許失敗臨界值]。

容忍失敗百分比

定義要容忍的失敗項目百分比。如果超過此值，則「地圖執行」會失敗。Step Functions 會計算失敗項目的百分比，因為失敗或逾時項目總數除以項目總數的結果。您必須指定一個介於零和 100 之間的值。預設百分比值為零，也就是說，如果工作流程的任何一個子工作流程執行失敗或逾時，工作流程就會失敗。如果將百分比指定為 100，即使所有子工作流程執行都失敗，工作流程也不會失敗。

或者，您可以將百分比指定為「分散式對映」狀態輸入中現有鍵值對的參考路徑。此路徑必須在執行階段解析為介於 0 到 100 之間的正整數。您可以在ToleratedFailurePercentagePath子欄位中指定參照路徑。

例如，假設有下列輸入：

```
{
  "percentage": 15
}
```

您可以使用該輸入的參考路徑來指定百分比，如下所示：

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
    "ToleratedFailurePercentagePath": "$.percentage"
    ...
  }
}
```

> **Important**
>
> 您可以在分散式地圖狀態定義中指定ToleratedFailurePercentage或ToleratedFailurePercentagePath，但不能同時指定兩者。

容忍失敗計數

定義要容忍的失敗項目數。如果超過此值，則「地圖執行」會失敗。

或者，您可以將 count 指定為分散式對映狀態輸入中現有鍵值對的參考路徑。此路徑必須在執行階段解析為正整數。您可以在ToleratedFailureCountPath子欄位中指定參照路徑。

例如，假設有下列輸入：

```
{
  "count": 10
}
```

您可以使用該輸入的參考路徑來指定數字，如下所示：

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
    "ToleratedFailureCountPath": "$.count"
    ...
  }
}
```

> **Important**
>
> 您可以在分散式地圖狀態定義中指定`ToleratedFailureCount`或`ToleratedFailureCountPath`，但不能同時指定兩者。

轉換

當您開始新的狀態機執行時，系統會從頂層`StartAt`欄位中參照的狀態開始。此欄位 (以字串形式指定) 必須完全相符 (包括大小寫) 在工作流程中的狀態名稱。

狀態執行之後，AWS Step Functions會使用`Next`欄位的值來決定下一個要前進的狀態。

`Next`欄位也會將狀態名稱指定為字串。此字串區分大小寫，且必須與狀態機器描述中指定的狀態名稱完全相符

例如，以下狀態包含對 `NextState` 的移轉。

```
"SomeState" : {
  ...,
  "Next" : "NextState"
}
```

大多數狀態只允許具有`Next`欄位的單一轉移規則。但是，某些流量控制狀態 (例如`Choice`狀態) 可讓您指定多個轉移規則，每個規則都有自己`Next`的欄位。Amazon 狀態語言提供每個您可以指定之狀態類型的詳細資訊，包括有關如何指定轉換的資訊。

狀態可以具有來自其他狀態的多個傳入轉換。

程序會重複執行，直到達終端機狀態 (具有、或的狀態`"End": true`) `"Type": Succeed`"Type": `Fail`，或發生執行階段錯誤為止。

當你執redrive行時，它被認為是一個狀態轉換。此外，在中重新執行的所有狀態也redrive會被視為狀態轉換。

下列規則適用於狀態機器內的狀態：

- 狀態可以在封閉區塊內以任何順序出現。但是，它們的列出順序不會影響它們的運行順序。該順序取決於狀態的內容。
- 在狀態機內，只能有一個狀態指定為狀`start`態。狀`start`態由頂層結構中`StartAt`欄位的值定義。
- 根據您的狀態機器邏輯（例如，如果狀態機具有多個邏輯分支），您可能會有多個end狀態。
- 如果您的狀態機只包含一種狀態，則它可以是開始和結束狀態。

分散式貼圖狀態下的轉移

當您在分散式模式中使用`Map`狀態時，會針對分散式地圖狀態啟動的每個子工作流程執行向您收取一次狀態轉換費用。當您在內嵌模式中使用`Map`狀態時，不會針對內嵌對應狀態的每次迭代向您收取狀態轉換費用。

您可以使用「分散式」模式中的`Map`狀態來最佳化成本，並在`Map`狀態定義中包含巢狀工作流程。當您啟動 Express 類型的子工作流程執行時，「分散式對應」狀態也會增加更多值。Step Functions 數會儲存 Express 子工作流程執行的回應和狀態，以減少在 CloudWatch 記錄中儲存執行資料的需求。您也可以存取「分散式對應」狀態下可用的流程控制項，例如定義錯誤閾值或批次處理項目群組。如需 Step Functions 定價的相關資訊，請參閱AWS Step Functions定價。

狀態機器資料

狀態機器資料的格式如下：

- 狀態機器的初始輸入
- 狀態之間傳遞的資料
- 來自狀態機器的輸出

本節將說明 AWS Step Functions 會如何格式化和使用狀態機器的資料。

主題

資料格式

狀態機數據由 JSON 文本表示。您可以使用 JSON 支援的任何資料類型，將值提供給狀態機器。

> **Note**
>
> - JSON 文字格式的數字符合JavaScript語意。這些數字通常對應到雙精度 IEEE-854 值。
> - 以下是有效的 JSON 文字：
>   - 獨立的引號分隔字串
>   - 物件
>   - 陣列
>   - 數字
>   - 布尔值
>   - `null`
> - 狀態的輸出會成為下一個狀態的輸入。但是，您可以使用「輸入和輸出處理」來限制狀態處理輸入資料的子集。

狀態機器輸入/輸出

您可以通過兩種方式之一將初始輸入數據提供給AWS Step Functions狀態機。您可以在開始執`StartExecution`行時將資料傳遞至動作。您也可以從 Step Functions 主控台將資料傳遞至狀態機器。初始資料會傳送至狀態機器的 `StartAt` 狀態。如果沒有提供輸入，預設會是空的物件 (`{}`)。

執行的輸出會由最後狀態 (`terminal`) 傳回。此輸出會在執行結果中以 JSON 文字呈現。

對於標準工`DescribeExecution`作流程，您可以使用外部呼叫者 (例如動作) 從執行歷程記錄擷取執行結果。您可以在「步驟函數」主控台上檢視執行結果。

對於 Express 工作流程，如果您啟用記錄功能，則可以從記CloudWatch錄擷取結果，或在 Step Functions 主控台中檢視和偵錯執行。如需詳細資訊，請參閱 記錄使用CloudWatch日誌 及 在 Step Functions 主控台上檢視和偵錯執行。

您還應該考慮與狀態機相關的配額。如需詳細資訊，請參閱 配額

狀態輸入/輸出

每個狀態的輸入皆包含先前狀態的 JSON 文字，或者針對 `StartAt` 狀態，則是要執行的輸入。某些流程控制狀態會將他們的輸入重複到他們的輸出中。

在以下範例中，狀態機器會同時新增兩個數字。

1. 定義 AWS Lambda 函數。

```
function Add(input) {
  var numbers = JSON.parse(input).numbers;
  var total = numbers.reduce(
    function(previousValue, currentValue, index, array) {
      return previousValue + currentValue; });
  return JSON.stringify({ result: total });
}
```

2. 定義 狀態機器。

```
{
  "Comment": "An example that adds two numbers together.",
  "StartAt": "Add",
  "Version": "1.0",
  "TimeoutSeconds": 10,
  "States":
    {
        "Add": {
          "Type": "Task",
          "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Add",
          "End": true
        }
    }
}
```

3. 開始執行以下 JSON 文字。

```
{ "numbers": [3, 4] }
```

狀Add態會接收 JSON 文字，並將其傳遞給 Lambda 函數。

Lambda 函數將計算結果返回到狀態。

狀態會在它的輸出中傳回以下值。

```
{ "result": 7 }
```

由於 Add 也是狀態機器中的最終狀態，此值會做為狀態機器的輸出傳回。

如果最終狀態沒有傳回輸出，則狀態機器會傳回空的物件 ({})。

如需詳細資訊，請參閱 Step Functions 中的輸入和輸出處理。

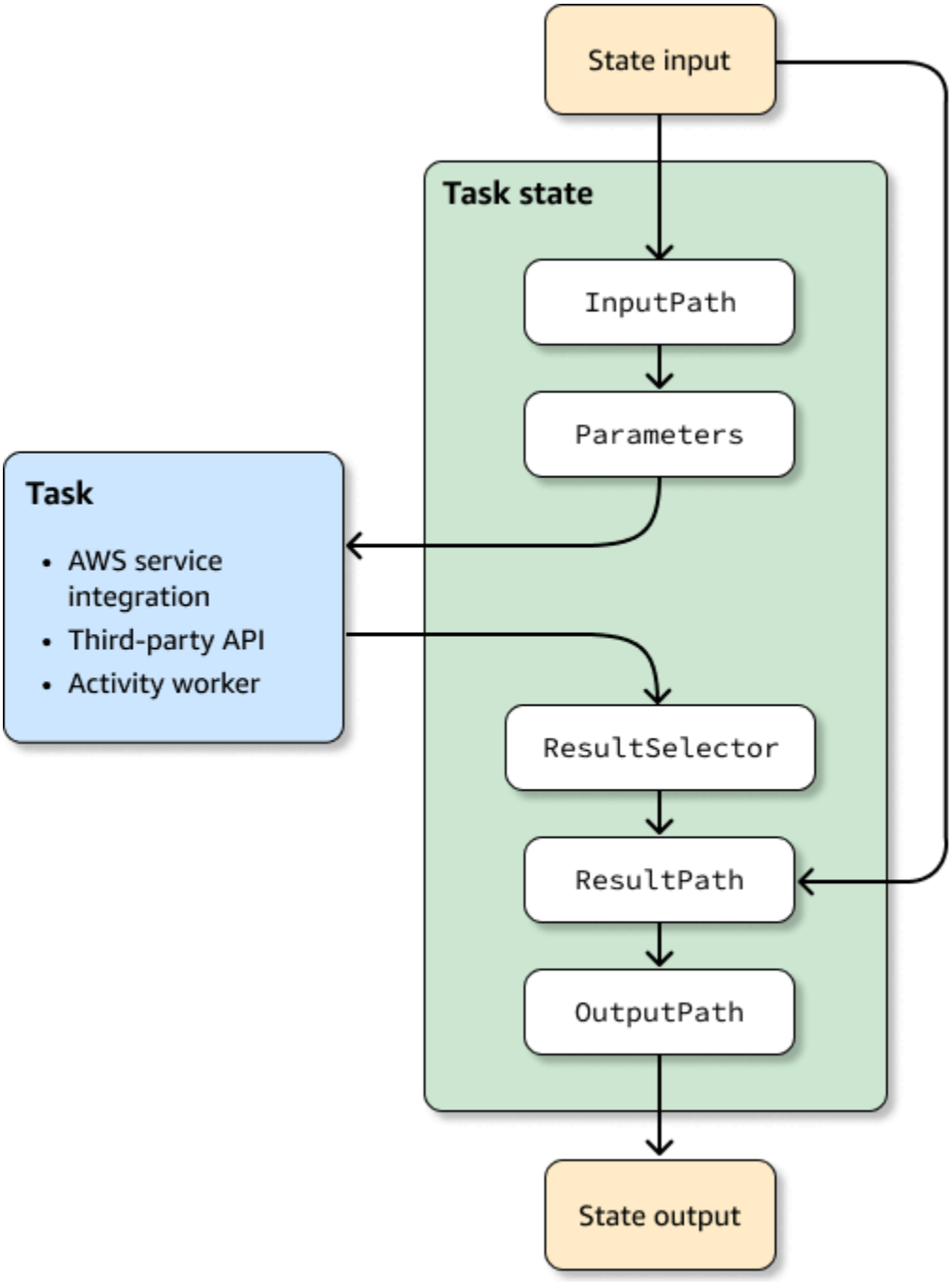
Step Functions 中的輸入和輸出處理

Step Functions 執行會接收 JSON 文字做為輸入，並將該輸入傳遞至工作流程中的第一個狀態。個別狀態會收到 JSON 做為輸入，而且通常會將 JSON 當作輸出傳遞至下一個狀態。了解此資訊如何在狀態之間傳送，以及了解如何篩選和操作此資料，可在 AWS Step Functions 中有效地設計和實作工作流程。

在 Amazon 州語言中，這些欄位篩選和控制 JSON 從一個狀態到另一個狀態的流程：

- `InputPath`
- `Parameters`
- `ResultSelector`
- `ResultPath`
- `OutputPath`

下圖顯示 JSON 資訊如何在工作狀態中移動。 `InputPath`選取要傳遞給`Task`狀態工作的 JSON 輸入部分 (例如 AWS Lambda 函數)。 `ResultPath`然後選擇要傳遞給輸出的狀態輸入和任務結果的組合。`OutputPath`可以過濾 JSON 輸出，以進一步限制傳遞給輸出的信息。



`InputPath`、、`Parameters`、`ResultSelector`、`ResultPath`和`OutputPath`每個 JSON 在工作流程中的每個狀態中移動時操作。

每一項都可以使用路徑，從輸入或結果中選取部分的 JSON。路徑是一個字符串，開頭$，標識 JSON 文本中的節點。Step Functions 路徑使用JsonPath語法。

> **Tip**
>
> 使用 Step Functions 主控台中的資料流程模擬器來測試 JSON 路徑語法、更好地瞭解如何在狀態內操作資料，並查看資料在狀態之間傳遞的方式。

> **Tip**
>
> 若要部署包含輸入和輸出處理的工作流程範例 AWS 帳戶，請參閱單元 6- AWS Step Functions 工作坊的輸入和輸出處理。

主題

路徑

在 Amazon 州語言中，路徑是一個字串，開頭$為可用來識別 JSON 文字中的元件。路徑遵循JsonPath語法。當指定 `InputPath`、`ResultPath` 和 `OutputPath` 的值時，您可以指定用來存取輸入子集的路徑。如需更多資訊，請參閱 Step Functions 中的輸入和輸出處理。

> **Note**
>
> 您也可以使用狀態定義中 `Parameters` 欄位的路徑，指定輸入或內容物件的 JSON 節點。請參閱 將參數傳遞至服務 API。

如果您的欄位名稱包含 JsonPath ABNF 規則`member-name-shorthand`定義中未包含的任何字元，則必須使用方括號標記法。因此，若要編碼特殊字元，例如標點符號 (排除_)，您必須使用方括號符號。例如 `$.abc.['def ghi']`。

參考路徑

參考路徑是一種語法有限的路徑，可供僅只識別 JSON 結構中的單一節點：

- 您可以僅使用點 (`.`) 和方括號 (`[ ]`) 符號來存取物件欄位。
- 不支援函數，例如 `length()`。
- 詞法運算符，這是非符號的，例如`subsetof`不支持。
- 不支援依規則運算式篩選或參考 JSON 結構中的其他值。
- 與篩選器中處理的目前節點相符的`@`運算子不符合純量值。它只匹配對象。

例如，如果狀態輸入資料包含下列值：

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  },
  "jar": [{"a": 1}, {"a": 5}, {"a": 2}, {"a": 7}, {"a": 3}]
}
```

下列參考路徑會傳回下列。

```
$.foo => 123
$.bar => ["a", "b", "c"]
$.car.cdr => true
$.jar[?(@.a >= 5)] => [{"a": 5}, {"a": 7}]
```

有些狀態使用路徑和參考路徑來控制狀態機器的流程，或設定狀態的設定或選項。如需詳細資訊，請參閱使用資料流程模擬器建模工作流程輸入和輸出路徑處理和在中有效使用 JSONPath。 AWS Step Functions

扁平化陣列的陣列

如果狀態機器中的平行或 Map state 傳回陣列陣列陣列，您可以將它們轉換成含有ResultSelector欄位的平面陣列。您可以將此欄位包含在「平行」或「對映」狀態定義中，以操作這些狀態的結果。

若要扁平化陣列，請在`ResultSelector`欄位中使用 JMESPath `[*]` 語法，如下列範例所示。

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

如需示範如何平面化陣列的範例，請參閱下列自學課程中的步驟 3：

- 使用 Lambda 函數處理整批資料
- 使用 Lambda 函數處理個別資料項目

InputPath、參數和 ResultSelector

Parameters和ResultSelector欄位提供了一種在 InputPath JSON 在工作流程中移動時操作的方法。 InputPath可以透過使用路徑篩選 JSON 標記法來限制傳遞的輸入 (請參閱路徑)。您可以使用 Parameters 欄位來傳遞索引鍵/值組集合，其中的值可能是您在狀態機器定義中的定義值，或是使用路徑從輸入所選取得出的值。該ResultSelector字段提供了一種在應用之前ResultPath操作狀態結果的方法。

AWS Step Functions 先套用InputPath欄位，然後套用Parameters欄位。您可以先使用 InputPath 將原始輸入篩選成您所要的選取範圍，然後套用 Parameters 來進一步操作輸入，或增加新的值。然後，您可以在套用之前ResultPath使用ResultSelector欄位來操作狀態的輸出。

> **Tip**
>
> 使用 Step Functions 主控台中的資料流程模擬器來測試 JSON 路徑語法、更好地瞭解如何在狀態內操作資料，並查看資料在狀態之間傳遞的方式。

InputPath

使用 InputPath 來選取一部分的狀態輸入。

例如，假設狀態的輸入包含下列項目：

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
```

```
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

您可以套用 InputPath。

```
"InputPath": "$.dataset2",
```

透過前述的 InputPath，依輸入方式傳遞的 JSON 將如下所示。

```
{
  "val1": "a",
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> 路徑可以產生一組值。請考量下列範例。
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> 如果您套用路徑 $.a[0:2]，則結果如下。
>
> ```
> [ 1, 2 ]
> ```

參數

本節說明使用「參數」欄位的不同方式。

鍵/值對

使用此Parameters欄位建立作為輸入傳遞的索引鍵值配對集合。每一組的值可以是您包含在狀態機器定義中的靜態值，或使用路徑從輸入或內容物件中所選取的值。對於使用路徑來選取值的索引鍵/值組，索引鍵名稱必須以 .$ 做為結尾。

例如，假設您提供以下輸入。

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

若要選取一些資訊，您可以在狀態機器定義中指定這些參數。

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

在前述輸入及 `Parameters` 欄位條件下，這是所傳遞的 JSON。

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

除了輸入，您可以存取特殊的 JSON 物件，即所謂的內容物件。內容物件包含關於狀態機器執行的資訊。請參閱 內容物件。

連線資源

Parameters 欄位也可以將資訊傳遞給已連線的資源。例如，如果您的任務狀態正在協調 AWS Batch 工作，則可以將相關 API 參數直接傳遞給該服務的 API 動作。如需詳細資訊，請參閱：

- 將參數傳遞至服務 API
- 使用其他 服務

Amazon S3

如果您在各個狀態之間傳遞的 Lambda 函數資料可能會成長到 262,144 個位元組以上，我們建議您使用 Amazon S3 存放資料，並實作下列其中一種方法：

- 在工作流程中使用分散式地圖狀態，以便Map狀態可以直接從 Amazon S3 資料來源讀取輸入。如需詳細資訊，請參閱 在分散式模式中使用對應狀態。
- 剖析Payload參數中儲存貯體的 Amazon 資源名稱 (ARN)，以取得值區名稱和金鑰值。如需詳細資訊，請參閱 使用 Amazon S3 ARN 而不是傳遞大型有效載荷。

或者，您可以調整實施以在執行中傳遞較小的有效載荷。

ResultSelector

在套用之前ResultPath，使用ResultSelector欄位來操作狀態的結果。此ResultSelector欄位可讓您建立索引鍵值組的集合，其中值為靜態或從狀態的結果中選取。使用該ResultSelector字段，您可以選擇要傳遞給該ResultPath字段的狀態結果的哪些部分。

> **Note**
>
> 使用該ResultPath字段，您可以將ResultSelector字段的輸出添加到原始輸入中。

ResultSelector是處於下列狀態的選擇性欄位：

- Map
- 任務
- 平行

例如，Step Functions 服務整合會傳回結果中有效負載以外的中繼資料。 ResultSelector可以選擇結果的部分，並將它們與狀態輸入合併ResultPath。在此範例中，我們只想選取resourceType和ClusterId，並將其與來自 Amazon EMR 建立叢集 .sync 的狀態輸入合併。鑑於以下內容：

```
{
  "resourceType": "elasticmapreduce",
  "resource": "createCluster.sync",
  "output": {
    "SdkHttpMetadata": {
      "HttpHeaders": {
        "Content-Length": "1112",
        "Content-Type": "application/x-amz-JSON-1.1",
        "Date": "Mon, 25 Nov 2019 19:41:29 GMT",
        "x-amzn-RequestId": "1234-5678-9012"
      },
      "HttpStatusCode": 200
    },
    "SdkResponseMetadata": {
      "RequestId": "1234-5678-9012"
    },
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

然後，您可以選擇resourceType並ClusterId使用ResultSelector：

```
"Create Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::elasticmapreduce:createCluster.sync",
  "Parameters": {
    <some parameters>
  },
  "ResultSelector": {
    "ClusterId.$": "$.output.ClusterId",
    "ResourceType.$": "$.resourceType"
  },
  "ResultPath": "$.EMROutput",
  "Next": "Next Step"
}
```

使用給定的輸入，使用ResultSelector產生：

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

扁平化陣列的陣列

如果狀態機器中的平行或 Map state 傳回陣列陣列陣列，您可以將它們轉換成含有ResultSelector欄位的平面陣列。您可以將此欄位包含在「平行」或「對映」狀態定義中，以操作這些狀態的結果。

若要扁平化陣列，請在ResultSelector欄位中使用 JMESPath [*] 語法，如下列範例所示。

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

如需示範如何平面化陣列的範例，請參閱下列自學課程中的步驟 3：

- 使用 Lambda 函數處理整批資料
- 使用 Lambda 函數處理個別資料項目

ResultPath

狀態的輸出可以是其輸入的複本、所產生的結果 (例如，從 Task 狀態之 Lambda 函數的輸出)，或輸入和結果的結合。使用 ResultPath 以控制哪些組合會傳遞至狀態輸出。

以下狀態類型可以產生結果，並可包含 ResultPath：

- Pass
- 任務
- 平行
- Map

使用 ResultPath 合併任務結果與任務輸入，或選取其中一個。您提供給 ResultPath 的路徑會控制哪些資訊會傳遞到輸出。

> **Note**
>
> `ResultPath` 僅限於使用 參考路徑，這會限制範圍以便僅識別 JSON 中的單一節點。請參考路徑參閱 Amazon 各州語言。

這些範例是以建立使用 Lambda 的 Step Functions 狀態機器教學課程中描述的狀態機器和 Lambda 函數為基礎。完成該教學，並透過在 `ResultPath` 欄位中嘗試不同路徑來測試不同輸出。

使用 ResultPath 來：

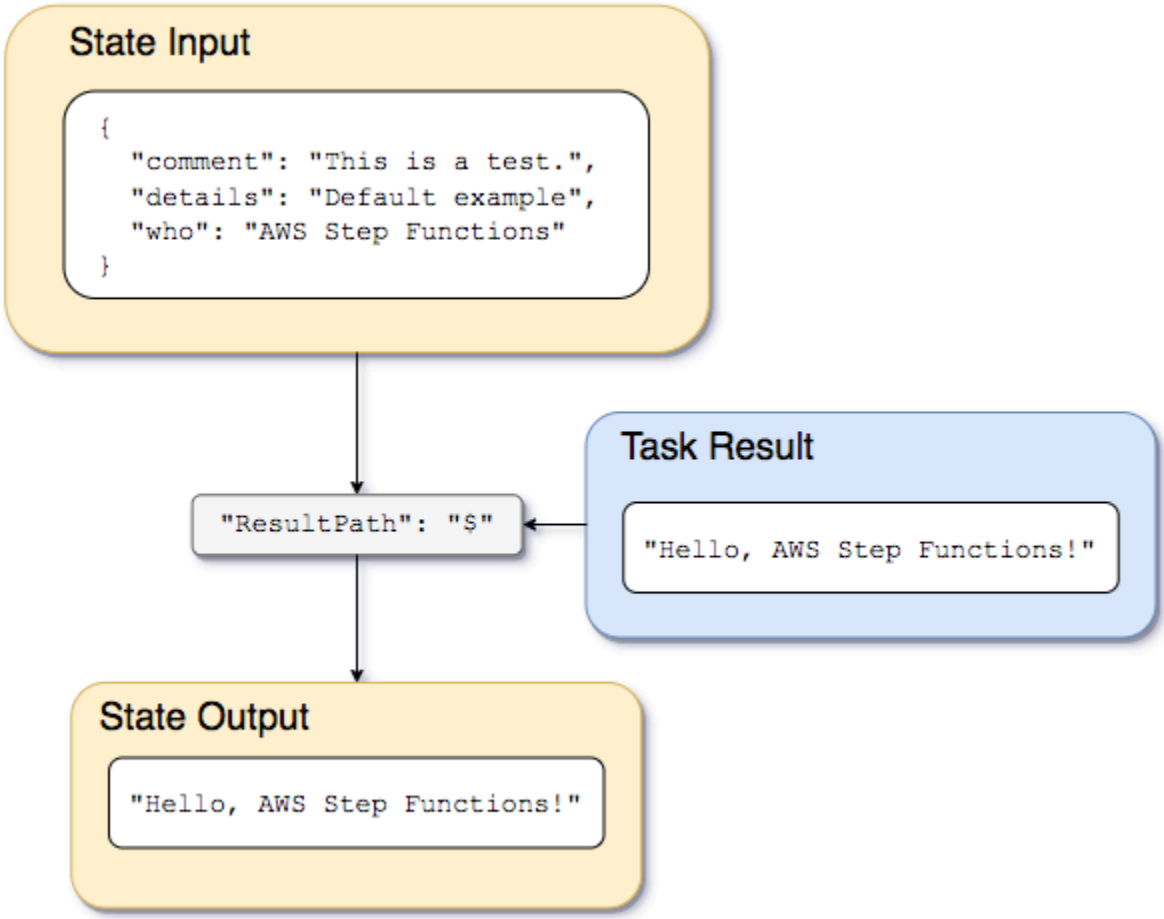
- 用 ResultPath 於將輸入取代為結果
- 捨棄結果並保留原始輸入
- 用 ResultPath 於將結果包含在輸入中
- 用 ResultPath 於使用結果更新輸入中的節點
- 用於 ResultPath 將錯誤和輸入都包含在 Catch

> **Tip**
>
> 使用 Step Functions 主控台中的資料流程模擬器來測試 JSON 路徑語法、更好地瞭解如何在狀態內操作資料，並查看資料在狀態之間傳遞的方式。

用 ResultPath 於將輸入取代為結果

如果您不指定 `ResultPath`，預設行為會視為您已指定 `"ResultPath": "$"`。由於這會告訴狀態以結果取代整個輸入，所以會以來自任務結果的結果完全取代狀態輸入。

下圖顯示 `ResultPath` 如何以任務結果完全取代輸入。



使用中所述的狀態機器和 Lambda 函數建立使用 Lambda 的 Step Functions 狀態機器，並將 Lambda 函數的服務整合類型變更為 AWS SDK 整合。若要這樣做，請在Task狀態Resource欄位中指定 Lambda 函數 Amazon 資源名稱 (ARN)，如下列範例所示。使用 AWS SDK 整合可確保Task狀態結果僅包含不含任何中繼資料的 Lambda 函數輸出。

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

然後，傳遞以下輸入：

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Lambda 函數會提供下列結果。

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> 您可以在Step Functions控制台上查看此結果。若要這麼做，請在主控台的 [執行詳細資訊] 頁面上，選擇 [圖形] 檢視中的Lambda函數。然後，選擇產量步驟詳情窗格中的選項卡以查看此結果。

如果ResultPath未在狀態中指定，或者如果"ResultPath": "$"已設定，則狀態的輸入會由 Lambda 函數的結果取代，且狀態的輸出如下所示。

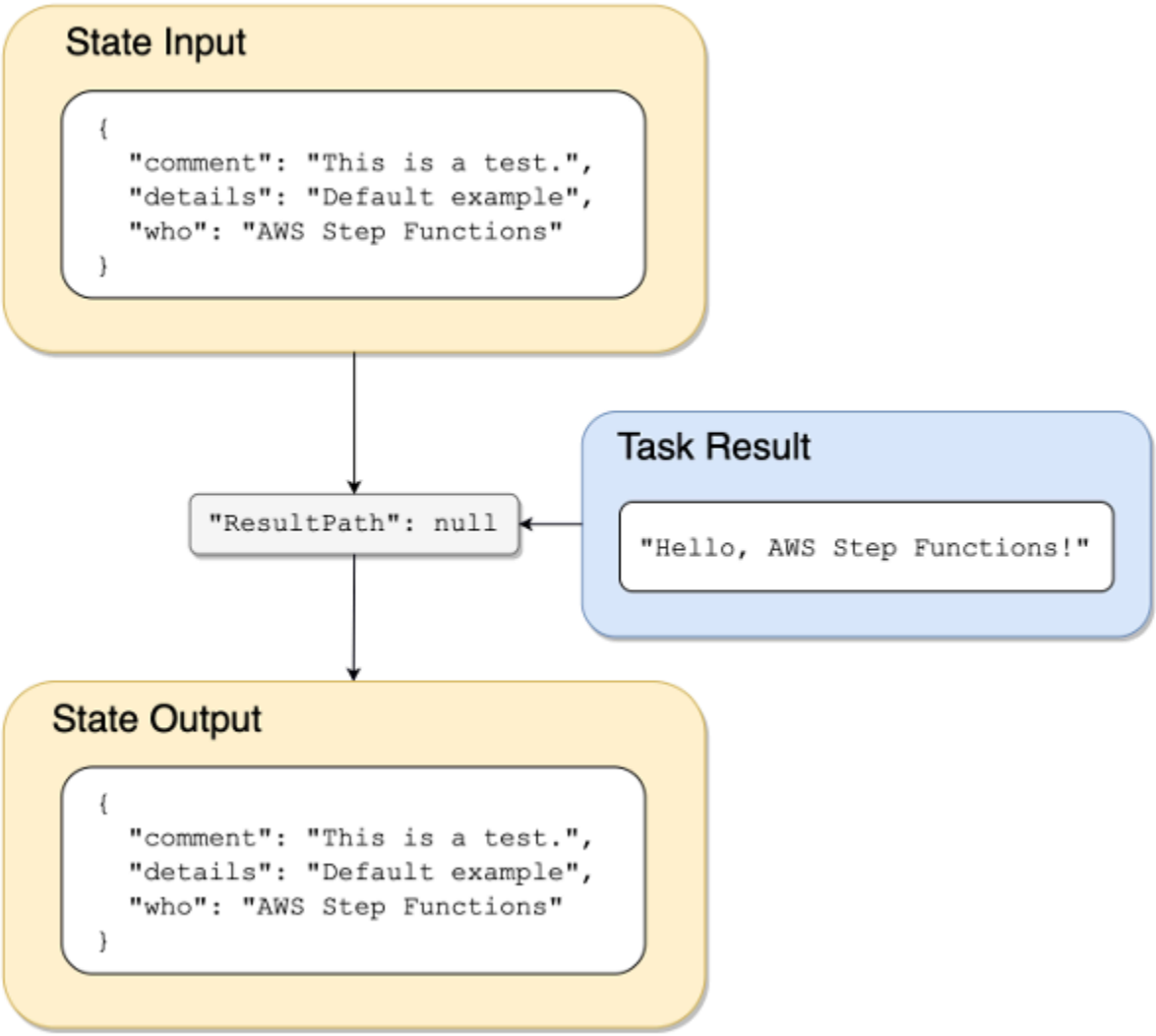
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> ResultPath 是在傳遞至輸出之前，用於在輸入中包含來自結果的內容。但是，如果未指定 ResultPath，預設會是取代整個輸入。

捨棄結果並保留原始輸入

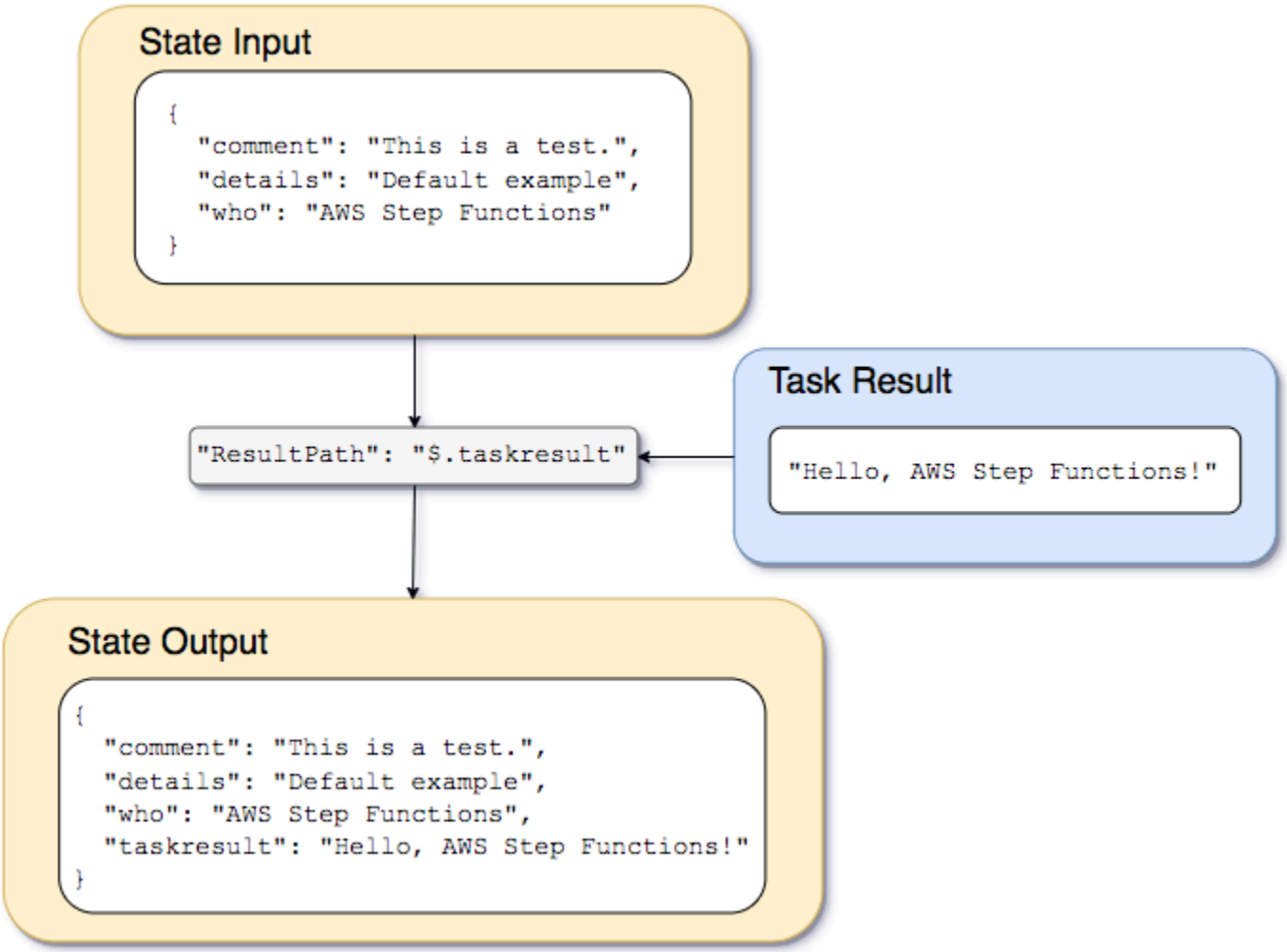
如果您將 ResultPath 設定為 null，其會將原始輸入傳遞給輸出。使用 "ResultPath": null 時，狀態的輸入承載會直接複製到輸出，而不考慮結果。

下圖顯示 Null ResultPath 會如何將輸入直接複製到輸出。



用 ResultPath 於將結果包含在輸入中

下圖顯示 `ResultPath` 如何在輸入中包含結果。



使用建立使用 Lambda 的 Step Functions 狀態機器教程中描述的狀態機和 Lambda 函數，我們可以傳遞以下輸入。

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Lambda 函數的結果如下。

```
"Hello, AWS Step Functions!"
```

為了保留輸入，插入 Lambda 函數的結果，然後將合併的 JSON 傳遞到下一個狀態，我們可以設置ResultPath為以下內容。

```
"ResultPath": "$.taskresult"
```

這包括 Lambda 函數與原始輸入的結果。

```
{
  "comment": "This is a test of input and output of a Task state.",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Lambda 函數的輸出會附加至原始輸入，做為的值taskresult。輸入 (包括新插入值) 會傳遞到下一個狀態。

您也可以在子節點的輸入中插入結果。將 ResultPath 設定為下列。

```
"ResultPath": "$.strings.lambdaresult"
```

使用以下輸入開始執行。

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

Lambda 函數的結果會插入為輸入中strings節點的子系。

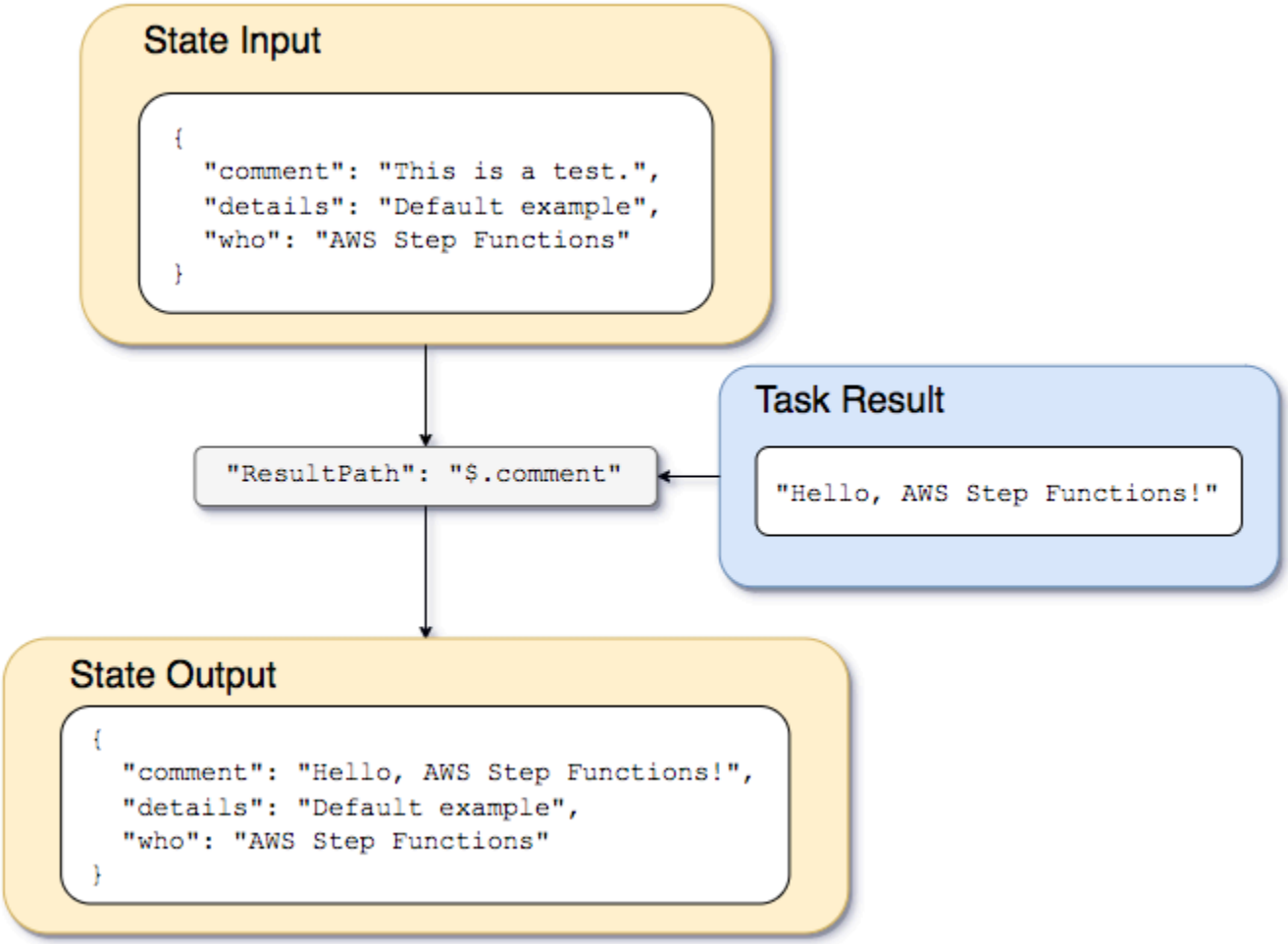
```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
```

```
    "string3": "baz",
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

狀態輸入現在會將原始輸入 JSON 做為子節點包含在結果中。

用 ResultPath 於使用結果更新輸入中的節點

下圖說明 `ResultPath` 如何以任務結果更新輸入中現有 JSON 節點的值。



使用建立使用 Lambda 的 Step Functions 狀態機器教程中描述的狀態機和 Lambda 函數的示例，我們可以傳遞以下輸入。

```
{
```

```
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Lambda 函數的結果如下。

```
Hello, AWS Step Functions!
```

因此，我們可以覆寫現有的節點，而不是保留輸入並將結果做為 JSON 中的新節點插入。

例如，就像省略或設定 "ResultPath": "$" 會覆寫整個節點一樣，您可以指定要使用結果覆寫的個別節點。

```
"ResultPath": "$.comment"
```

由於comment節點已存在於狀態輸入中，因此設ResultPath定"$.comment"會以 Lambda 函數的結果取代輸入中的該節點。如果沒有使用 OutputPath 進一步篩選，則將以下會傳遞至輸出。

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

"Hello, AWS Step Functions!"在狀態輸出中"This is a test of the input and output of a Task state."，comment節點的值會由 Lambda 函數的結果取代。

用於 ResultPath 將錯誤和輸入都包含在 **Catch**

使用 Step Functions 狀態機處理錯誤條件 教學示範如何使用狀態機器截獲錯誤。在某些情況下，您可能想要保留原始輸入的錯誤。在 Catch 中使用 ResultPath 以包含錯誤與原始輸入，而不是將其取代。

```
"Catch": [{
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

如果之前的 Catch 陳述式截獲錯誤，它會在狀態輸入內的 error 節點包含結果。例如，透過以下輸入：

```
{"foo": "bar"}
```

在截獲錯誤時的狀態輸出如下。

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

如需錯誤處理的詳細資訊，請參閱以下內容：

- Step Functions 中的錯誤處理
- 使用 Step Functions 狀態機處理錯誤條件

OutputPath

OutputPath 可讓您選擇一部分的狀態輸出，傳遞至下一個狀態。這可讓您篩選掉不需要的資訊，只傳遞您關注的 JSON 部分。

如未指定 OutputPath，預設值將會是 $。這樣會將整個 JSON 節點 (由狀態輸入、任務結果及 ResultPath 決定) 傳遞至下一個狀態。

> **Tip**
>
> 使用 Step Functions 主控台中的資料流程模擬器來測試 JSON 路徑語法、更好地瞭解如何在狀態內操作資料，並查看資料在狀態之間傳遞的方式。

如需詳細資訊，請參閱下列內容：

- Amazon 州語言的路徑
- InputPath、ResultPath、與OutputPath範例
- 將靜態 JSON 作為參數傳遞

- Step Functions 中的輸入和輸出處理

InputPath、ResultPath、與OutputPath範例

狀態或狀態以外的Fail任何Succeed狀態都可以包括輸入和輸出處理欄位InputPath，例如ResultPath、或OutputPath。此外，等候和Choice狀態不支援ResultPath欄位。透過這些欄位，您可以使用 a JsonPath來篩選 JSON 資料在工作流程中移動時。

您也可以使用Parameters欄位，在 JSON 資料在工作流程中移動時操作。如需使用 Parameters 的詳細資訊，請參閱 InputPath、參數和 ResultSelector。

例如，從建立使用 Lambda 的 Step Functions 狀態機器教學課程中描述的AWS Lambda函數和狀態機開始。修改狀態機器，使其包含下列 InputPath、ResultPath 及 OutputPath。

```
{
  "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
      "InputPath": "$.lambda",
      "ResultPath": "$.data.lambdaresult",
      "OutputPath": "$.data",
      "End": true
    }
  }
}
```

使用以下輸入開始執行。

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
```

```
  }
}
```

假設comment和extra節點可以捨棄，但我們想要包含 Lambda 函數的輸出，並保留data節點中的資訊。

在已更新的狀態機器中，Task 狀態會更改為處理任務輸入。

```
"InputPath": "$.lambda",
```

狀態機器定義中的這一行會將任務輸入限制為僅限任務輸入中的 lambda 節點。拉姆達函數只接收 JSON 對象{"who": "AWS Step Functions"}作為輸入。

```
"ResultPath": "$.data.lambdaresult",
```

這ResultPath告訴狀態機將 Lambda 函數的結果插入到名為的節點中lambdaresult，作為原始狀態機輸入中data節點的子節點。因為我們沒有對原始輸入和使用的結果執行任何其他操作OutputPath，所以狀態的輸出現在包括 Lambda 函數的結果與原始輸入。

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

但是，我們的目標是只保留data節點，並包含 Lambda 函數的結果。 OutputPath過濾此組合的 JSON，然後再將其傳遞給狀態輸出。

```
"OutputPath": "$.data",
```

這只會選取原始輸入中要傳遞到輸出的 data 節點 (包括 ResultPath 所插入的 lambdaresult 子系)。狀態輸出會篩選到下列。

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

在此 Task 狀態中：

1. InputPath只會將lambda節點從輸入傳送至 Lambda 函數。
2. ResultPath 會插入結果做為原始輸入中 data 節點的子節點。
3. OutputPath過濾狀態輸入（現在包括 Lambda 函數的結果），以便僅將data節點傳遞給狀態輸出。

Example 操作原始狀態機輸入，結果和最終輸出 JsonPath

考慮下面的狀態機來驗證保險申請人的身份和地址。

> **Note**
>
> 若要檢視完整範例，請參閱步驟函式中的如何使用 JSON 路徑。

```
{
  "Comment": "Sample state machine to verify an applicant's ID and address",
  "StartAt": "Verify info",
  "States": {
    "Verify info": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Verify identity",
          "States": {
            "Verify identity": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
identity:$LATEST"
```

```
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Verify address",
          "States": {
            "Verify address": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
address:$LATEST"
              },
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

如果您使用下列輸入來執行此狀態機器，則執行會失敗，因為執行驗證的 Lambda 函數只需要將需要驗證的資料做為輸入。因此，您必須使用適當的指定包含要驗證之資訊的節點JsonPath。

```
{
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    },
    "interests": [
```

```
      {
        "category": "home",
        "type": "own",
        "yearBuilt": 2004
      },
      {
        "category": "boat",
        "type": "snowmobile",
        "yearBuilt": 2020
      },
      {
        "category": "auto",
        "type": "RV",
        "yearBuilt": 2015
      },
    ]
  }
}
```

若要指定 *check-identity* Lambda 函數必須使用的節點，請依下列方式使用InputPath欄位：

```
"InputPath": "$.data.identity"
```

若要指定 *check-address* Lambda 函數必須使用的節點，請依下列方式使用InputPath欄位：

```
"InputPath": "$.data.address"
```

現在，如果要將驗證結果存儲在原始狀態機輸入中，請使用該ResultPath字段，如下所示：

```
"ResultPath": "$.results"
```

但是，如果您只需要身份和驗證結果並丟棄原始輸入，請按如下方式使用該OutputPath字段：

```
"OutputPath": "$.results"
```

如需詳細資訊，請參閱 Step Functions 中的輸入和輸出處理。

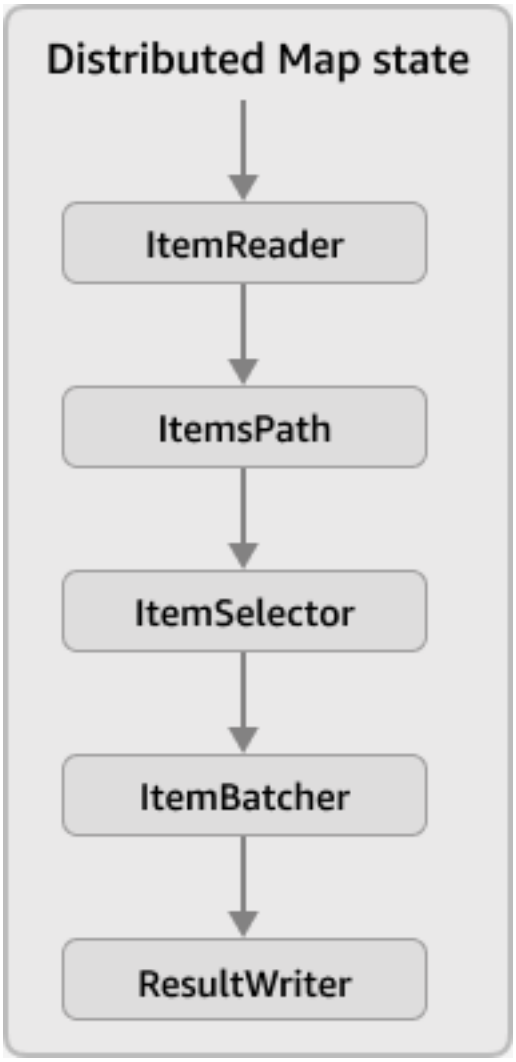
映射狀態輸入和輸出字段

Map狀態會同時重複執行資料集中的項目集合，例如 JSON 陣列、Amazon S3 物件清單或 Amazon S3 儲存貯體中 CSV 檔案的資料列。它為集合中的每個項目重複一組步驟。您可以使用這些欄位來設

定Map狀態接收的輸入及其產生的輸出。「步驟函數」會依照下列清單和插圖中所示的順序，套用「分散式對應」狀態中的每個欄位：

> **Note**
>
> 根據您的使用案例，您可能不需要套用所有這些欄位。

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher
5. ResultWriter



> **Note**
>
> 在 Step Functions 主控台的資料流程模擬器中，目前無法使用這些Map狀態輸入和輸出欄位。

ItemReader

該ItemReader字段是一個 JSON 對象，它指定一個數據集和它的位置。一個分散式地圖狀態使用此資料集作為其輸入。下列範例顯示的語法ItemReader如果您的資料集是存放在 Amazon S3 儲存貯體中的 CSV 檔案，請按照欄位。

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
```

```
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> 在工作流程 Studio 中，您可以指定資料集及其位置項目來源欄位。

目錄

本公司的內容 ItemReader 領域

視您的資料集而定，`ItemReader`欄位各不相同。例如，如果您的資料集是從工作流程的上一個步驟傳遞的 JSON 陣列，`ItemReader`省略字段。如果您的資料集是 Amazon S3 資料來源，則此欄位包含下列子欄位。

`ReaderConfig`

指定下列詳細資訊的 JSON 物件：

- `InputType`

  指定 Amazon S3 資料來源的類型，例如 CSV 檔案、物件、JSON 檔案或 Amazon S3 庫存清單。在工作流程 Studio 中，您可以從Amazon S3 項目來源下面的下拉列表項目來源欄位。

- `CSVHeaderLocation`

  > **Note**
  >
  > 只有在使用 CSV 檔案作為資料集時，才必須指定此欄位。

接受下列其中一個值來指定欄標題的位置：

> **Important**
>
> 目前，Step Functions 支援高達 10 KB 的 CSV 標頭。

- `FIRST_ROW`— 如果檔案的第一行是標頭，請使用此選項。
- `GIVEN`— 使用此選項可指定狀態機器定義內的標頭。例如，您的 CSV 檔案包含以下資料。

```
1,307,3.5,1256677221
1,481,3.5,1256677456
1,1091,1.5,1256677471
...
```

提供下列 JSON 陣列做為 CSV 標頭。

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> 在工作流程工作室中，您可以在下找到此選項其他配置在項目來源欄位。

- `MaxItems`

限制傳送至Map狀態。例如，假設您提供的 CSV 檔案包含 1000 個資料列，並且指定的限制為 100。然後，解釋器通過只要一百個資料列到Map狀態。該Mapstate 按順序處理項目，從標題行之後開始。

預設情況下，Mapstate 遍歷指定數據集中的所有項目。

> **Note**
>
> 目前，您可以指定最多 100 萬的限制。該分散式地圖狀態停止讀取超出此限制的項目。

> **Tip**
>
> 在工作流程工作室中，您可以在下找到此選項其他配置在項目來源欄位。

您可以指定參考路徑到您的現有鍵值對分散式地圖狀態輸入。此路徑必須解析為正整數。您可以在`MaxItemsPath`子欄位。

> **Important**
>
> 您可以指定`MaxItems`或`MaxItemsPath`子欄位，但不能同時使用兩者。

`Resource`

Step Functions 式必須根據指定的資料集叫用的 Amazon S3 API 動作。

`Parameters`

JSON 物件，指定資料集所在的 Amazon S3 儲存貯體名稱和物件金鑰。

> **Important**
>
> 確保您的 Amazon S3 存儲桶位於相同AWS 帳戶和AWS 區域作為您的狀態機。

資料集範例

您可以用下列其中一個選項指定為您的資料集：

- 上一步中的 JSON 數組
- Amazon S3 對象列表
- Amazon S3 存儲桶中的 JSON 文件
- Amazon S3 儲存貯體中的 CSV 檔案
- Amazon S3 清查清單

> **Important**
>
> Step Functions 需要適當的許可才能存取您使用的 Amazon S3 資料集。如需資料集的 IAM 政策的相關資訊，請參閱資料集適用的 IAM 政策。

上一步中的 JSON 數組

一個分散式地圖狀態可以在工作流程中接受從上一步驟傳遞的 JSON 輸入。此輸入必須是陣列，或者必須包含特定節點內的陣列。若要選取包含陣列的節點，您可以用`ItemsPath`欄位。

若要處理陣列中的個別項目，分散式地圖狀態啟動每個陣列項目的子工作流程執行。下列索引標籤顯示傳遞至Mapstate 和子工作流程執行的對應輸入。

> **Note**
>
> Step Functions 省略`ItemReader`當您的資料集是上一個步驟的 JSON 陣列時的欄位。

Input passed to the Map state

考慮下面的三個項目的 JSON 數組。

```
"facts": [
    {
        "verdict": "true",
        "statement_date": "6/11/2008",
        "statement_source": "speech"
    },
    {
        "verdict": "false",
        "statement_date": "6/7/2022",
```

```
        "statement_source": "television"
    },
    {
        "verdict": "mostly-true",
        "statement_date": "5/18/2016",
        "statement_source": "news"
    }
]
```

Input passed to a child workflow execution

該分散式地圖狀態啟動三個子工作流程執行。每次執行都會接收一個陣列項目作為輸入。下列範例顯示子工作流程執行所接收的輸入。

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Amazon S3 對象示例

一個分散式地圖狀態可以在 Amazon S3 儲存貯體中進行迭代。當工作流程執行到達Map狀態，Step Functions 調用ListObjectsV2API 動作，此動作會傳回 Amazon S3 物件中繼資料的陣列。在此陣列中，每個項目都包含資料，例如ETag和關鍵，用於儲存在值區中的資料。

若要處理陣列中的個別項目，分散式地圖狀態啟動子工作流程執行。例如，假設您的 Amazon S3 儲存貯體包含 100 個圖像。然後，調用後返回數組ListObjectsV2API 動作包含 100 個項目。該分散式地圖狀態然後啟動 100 個子工作流程執行，以處理每個陣列項目。

> **Note**
>
> - 目前，Step Functions 也包含您使用 Amazon S3 主控台在 Amazon S3 儲存貯體中建立資料夾的項目。這會導致額外的子工作流程執行由分散式地圖狀態。若要避免為資料夾建立額外的子工作流程執行，建議您使用AWS CLI以建立資料夾。如需詳細資訊，請參閱高級 Amazon S3 命令在AWS Command Line Interface用戶指南。
> - Step Functions 需要適當的許可才能存取您使用的 Amazon S3 資料集。如需資料集的 IAM 政策的相關資訊，請參閱資料集適用的 IAM 政策。

下列索引標籤顯示ItemReader欄位語法和輸入傳遞給此資料集的子工作流程執行。

ItemReader syntax

在此範例中，您已將資料 (包括影像、JSON 檔案和物件) 整理在名為的前置詞內processData在一個名為的 Amazon S3 存儲桶myBucket。

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

該分散式地圖狀態啟動與 Amazon S3 儲存貯體中存在的項目數量相同的子工作流程執行數量。下列範例顯示子工作流程執行所接收的輸入。

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

Amazon S3 存儲桶中的 JSON 文件

一個分散式地圖狀態可以接受存放在 Amazon S3 儲存貯體中的 JSON 檔案作為資料集。JSON 文件必須包含一個數組。

當工作流程執行到達Map狀態，Step Functions 調用GetObject用於擷取指定 JSON 檔案的 API 動作。該Mapstate 接著會重複執行陣列中的每個項目，並針對每個項目啟動子工作流程執行。例如，如果您的 JSON 檔案包含 1000 個陣列項目，Map狀態會啟動 1000 個子工作流程執行。

> **Note**
>
> - 用於啟動子工作流程執行的執行輸入不能超過 256 KB。但是，如果您隨後應用可選項，則 Step Functions 支持從 CSV 或 JSON 文件中讀取最多 8 MB 的項目`ItemSelector`欄位以減少項目的大小。
> - 目前，Step Functions 支援 10 GB 作為 Amazon S3 庫存報告中個別檔案的大小上限。不過，如果每個個別檔案小於 10 GB，則 Step Functions 可以處理超過 10 GB。
> - Step Functions 需要適當的許可才能存取您使用的 Amazon S3 資料集。如需資料集的 IAM 政策的相關資訊，請參閱資料集適用的 IAM 政策。

下列索引標籤顯示`ItemReader`欄位語法和輸入傳遞給此資料集的子工作流程執行。

在此範例中，假設您有名為的 JSON 檔案*`factcheck.json`*。您已將此檔案儲存在名為的前置字元內*`jsonDataset`*在 Amazon S3 存儲桶。以下是 JSON 資料集的範例。

```
[
  {
    "verdict": "true",
    "statement_date": "6/11/2008",
    "statement_source": "speech"
  },
  {
    "verdict": "false",
    "statement_date": "6/7/2022",
    "statement_source": "television"
  },
  {
    "verdict": "mostly-true",
    "statement_date": "5/18/2016",
    "statement_source": "news"
  },
  ...
]
```

ItemReader syntax

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:getObject",
    "ReaderConfig": {
```

```
        "InputType": "JSON"
    },
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "jsonDataset/factcheck.json"
    }
}
```

Input to a child workflow execution

該分散式地圖狀態啟動與 JSON 檔案中存在的陣列項目數量相同的子工作流程執行數量。下列範例顯示子工作流程執行所接收的輸入。

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Amazon S3 儲存貯體中的 CSV 檔案

一個分散式地圖狀態可以接受存放在 Amazon S3 儲存貯體中的 CSV 檔案作為資料集。如果您使用 CSV 檔案做為資料集，則必須指定 CSV 欄標題。如需如何指定 CSV 標頭的相關資訊，請參閱本公司的內容 ItemReader 領域。

因為沒有標準化的格式來建立和維護 CSV 檔案中的資料，因此「Step Functions 數」會根據下列規則剖析 CSV 檔案：

- 逗號 (,) 是用來分隔個別欄位的分隔符號。
- 換行符是分隔各個記錄的分隔符。
- 欄位會被視為字串。若要進行資料類型轉換，請使用States.StringToJson內在函數ItemSelector。
- 括住字串不需要雙引號 (「」)。但是，以雙引號括住的字串可以包含逗號和換行符，而不會用作分隔符號。
- 通過重複它們來逃避雙引號。
- 如果資料列中的欄位數目小於標頭中的欄位數目，Step Functions 會為遺漏值提供空字串。
- 如果列中的欄位數目大於標題中的欄位數目，則「Step Functions 數」會略過其他欄位。

如需有關 Step Functions 如何剖析 CSV 檔案的資訊，請參閱Example of parsing an input CSV file。

當工作流程執行到達Map狀態，Step Functions 調用GetObject用於擷取指定 CSV 檔案的 API 動作。該Mapstate 接著會重複執行 CSV 檔案中的每一列，並啟動子工作流程執行，以處理每一列中的項目。例如，假設您提供的 CSV 檔案包含 100 列作為輸入。然後，解釋器將每一行傳遞給Map狀態。該Mapstate 按序列處理項目，從標題列之後開始。

> **Note**
>
> - 用於啟動子工作流程執行的執行輸入不能超過 256 KB。但是，如果您隨後應用可選項，則 Step Functions 支持從 CSV 或 JSON 文件中讀取最多 8 MB 的項目`ItemSelector`欄位以減少項目的大小。
> - 目前，Step Functions 支援 10 GB 作為 Amazon S3 庫存報告中個別檔案的大小上限。不過，如果每個個別檔案小於 10 GB，則 Step Functions 可以處理超過 10 GB。
> - Step Functions 需要適當的許可才能存取您使用的 Amazon S3 資料集。如需資料集的 IAM 政策的相關資訊，請參閱資料集適用的 IAM 政策。

下列索引標籤顯示`ItemReader`欄位語法和輸入傳遞給此資料集的子工作流程執行。

ItemReader syntax

例如，假設您有名為的 CSV 檔案*ratings.csv*。然後，您已經將此文件存儲在名為前綴*csvDataset*在 Amazon S3 存儲桶。

```
{
  "ItemReader": {
    "ReaderConfig": {
      "InputType": "CSV",
      "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
      "Bucket": "myBucket",
      "Key": "csvDataset/ratings.csv"
    }
  }
}
```

Input to a child workflow execution

該分散式地圖狀態啟動與 CSV 檔案中存在的列數一樣多的子工作流程執行，但不包括標題列 (如果在檔案中)。下列範例顯示子工作流程執行所接收的輸入。

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
  "timestamp": "1256677221"
}
```

S3 庫存範例

一個分散式地圖狀態可以在 Amazon S3 儲存貯體中接受作為資料集。

當工作流程執行到達Map狀態，Step Functions 調用GetObject用於擷取指定的 Amazon S3 庫存資訊清單檔案的 API 動作。該Map然後，狀態重複執行詳細目錄中的物件，以傳回 Amazon S3 庫存物件中繼資料的陣列。

> **Note**
>
> - 目前，Step Functions 支援 10 GB 作為 Amazon S3 庫存報告中個別檔案的大小上限。不過，如果每個個別檔案小於 10 GB，則 Step Functions 可以處理超過 10 GB。
> - Step Functions 需要適當的許可才能存取您使用的 Amazon S3 資料集。如需資料集的 IAM 政策的相關資訊，請參閱資料集適用的 IAM 政策。

以下是 CSV 格式的庫存檔案範例。此檔案包括名為的物件csvDataset和imageDataset，這些儲存在名為的 Amazon S3 儲存貯體sourceBucket。

```
"sourceBucket","csvDataset/","0","2022-11-16T00:27:19.000Z"
"sourceBucket","csvDataset/titles.csv","3399671","2022-11-16T00:29:32.000Z"
"sourceBucket","imageDataset/","0","2022-11-15T20:00:44.000Z"
"sourceBucket","imageDataset/n02085620_10074.jpg","27034","2022-11-15T20:02:16.000Z"
...
```

> **Important**
>
> 目前，Step Functions 不支援使用者定義的 Amazon S3 庫存報告作為資料集。您也必須確定 Amazon S3 庫存報告的輸出格式為 CSV。如需 Amazon S3 庫存以及如何設定的詳細資訊，請參閱Amazon S3 清查在Amazon S3 使用者指南。

下列詳細目錄資訊清單檔案範例顯示詳細目錄物件中繼資料的 CSV 標頭。

```
{
  "sourceBucket" : "sourceBucket",
  "destinationBucket" : "arn:aws:s3:::inventory",
  "version" : "2016-11-30",
  "creationTimestamp" : "1668560400000",
  "fileFormat" : "CSV",
  "fileSchema" : "Bucket, Key, Size, LastModifiedDate",
  "files" : [ {
    "key" : "source-bucket/destination-prefix/
data/20e55de8-9c21-45d4-99b9-46c732000228.csv.gz",
    "size" : 7300,
    "MD5checksum" : "a7ff4a1d4164c3cd55851055ec8f6b20"
  } ]
}
```

下列索引標籤顯示ItemReader欄位語法和輸入傳遞給此資料集的子工作流程執行。

ItemReader syntax

```
{
  "ItemReader": {
    "ReaderConfig": {
      "InputType": "MANIFEST"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
      "Bucket": "destinationBucket",
      "Key": "destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/
manifest.json"
    }
  }
}
```

Input to a child workflow execution

```
{
  "LastModifiedDate": "2022-11-16T00:29:32.000Z",
  "Bucket": "sourceBucket",
  "Size": "3399671",
  "Key": "csvDataset/titles.csv"
}
```

根據您在設定 Amazon S3 庫存報告時選取的欄位，`manifest.json`檔案可能與顯示的範例有所不同。

資料集適用的 IAM 政策

使用 Step Functions 主控台建立工作流程時，Step Functions 可以根據工作流程定義中的資源自動產生 IAM 政策。這些原則包括允許狀態機器角色呼叫`StartExecution`用於分散式地圖狀態。這些原則也包含存取所需的最低權限 Step FunctionsAWS資源，例如 Amazon S3 儲存貯體和物件，以及 Lambda 函數。我們強烈建議您僅在 IAM 政策中加入必要的許可。例如，如果您的工作流程包含`Map`在分散式模式下說明，將您的政策範圍縮減到包含資料集的特定 Amazon S3 儲存貯體和資料夾。

> **Important**
>
> 如果您指定 Amazon S3 儲存貯體和物件，或前置詞參考路徑到您的現有鍵值對分散式地圖狀態輸入，請務必更新您的工作流程的 IAM 政策。將政策範圍縮小到值區和路徑在執行時期解析為的物件名稱。

下列 IAM 政策範例授予存取 Amazon S3 資料集所需的最低權限ListObjectsV2和GetObjectAPI 動作。

Example 適用於作為資料集的 Amazon S3 物件的 IAM 政策

下列範例顯示一項 IAM 政策，該政策授與存取其中組織之物件的最少權限*`processImages`*在一個名為的 Amazon S3 存儲桶*`myBucket`*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example 將 CSV 檔案作為資料集的 IAM 政策

以下範例顯示一項 IAM 政策，該政策授予存取名為的 CSV 檔案的最少權限*ratings.csv*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example 用於作為資料集的 Amazon S3 庫存的 IAM 政策

下列範例顯示授與存取 Amazon S3 庫存報告的最少權限的 IAM 政策。

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

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-
MMZ/manifest.json",
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/data/*"
            ]
        }
    ]
}
```

ItemsPath

使用ItemsPath欄位在提供給Map狀態的 JSON 輸入內選取陣列。狀Map態會針對陣列中的每個項目重複一組步驟。默認情況下，Map狀態設置ItemsPath為$，選擇整個輸入。如果Map狀態的輸入是 JSON 陣列，它會針對陣列中的每個項目執行迭代，並將該項目作為輸入傳遞給迭代。

> **Note**
>
> 只有在工作流程ItemsPath中使用從先前狀態傳遞的 JSON 輸入時，才能在「分散式對應」狀態中使用。

您可以使用ItemsPath欄位來指定輸入中指向用於反覆運算的 JSON 陣列的位置。的值ItemsPath必須是參考路徑，且該路徑必須指向 JSON 陣列。例如，請考慮包含兩個陣列的 Map 狀態輸入，如下列範例所示。

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
```

```
  "ThingsGiantsSay": [
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
    },
    {
      "say": "Fo!"
    },
    {
      "say": "Fum!"
    }
  ]
}
```

在這種情況下，您可以通過使用選擇來指定要用於Map狀態迭代的數組ItemsPath。下列狀態機器定義會使用 ItemsPath lt 指定輸入中的ThingsPiratesSay陣列，然後針對陣ThingsPiratesSay列中的每個項目執行SayWord傳遞狀態的迭代。

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```

處理輸入時，狀Map態適用於ItemsPath之後InputPath。它在InputPath過濾輸入後的有效輸入到狀態上進行操作。

如需 Map 狀態的詳細資訊，請參閱下列內容：

- 地圖狀態
- 對映狀態處理模式
- 使用內嵌對應狀態重複動作
- 內聯Map狀態輸入和輸出處理

ItemSelector

根據預設，狀態的有效輸入是存在於原始Map狀態輸入中的一組個別資料項目。此ItemSelector欄位可讓您在資料項目傳遞至Map狀態之前覆寫資料項目的值。若要覆寫值，請指定包含索引鍵值組集合的有效 JSON 輸入。這些配對可以是狀態機器定義中提供的靜態值、使用路徑從狀態輸入中選取的值，或是從上下文物件存取的值。

如果您使用路徑或上下文對象指定鍵值對，則鍵名必須以結尾。.$

> **Note**
>
> 此ItemSelector欄位會取代Map狀態內的Parameters欄位。如果您使用狀Map態定義中的Parameters欄位來建立自訂輸入，我們強烈建議您將它們取代為ItemSelector。

您可以在 ItemSelector「內嵌對應」狀態和「分散式地圖」狀態中指定欄位。

例如，請考慮下列 JSON 輸入，其中包含imageData節點內三個項目的陣列。對於每個*Map*狀態迭代，數組項被傳遞給迭代作為輸入。

```
[
  {
    "resize": "true",
    "format": "jpg"
  },
  {
    "resize": "false",
    "format": "png"
  },
  {
    "resize": "true",
    "format": "jpg"
  }
```

```
]
```

您可以使用ItemSelector欄位定義自訂 JSON 輸入，以覆寫原始輸入，如下列範例所示。步驟函數然後將此自定義輸入傳遞給每個*Map*狀態迭代。自訂輸入包含Map狀態的靜態值size和上下文物件資料的值。$$.Map.Item.Value上下文對象包含每個單獨的數據項的值。

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
}
```

下面的例子顯示了由內聯映射狀態的一個迭代接收的輸入：

```
{
  "size": 10,
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> 如需使用ItemSelector欄位的「分散式貼圖」狀態的完整範例，請參閱開始使用分散式地圖狀態。

ItemBatcher

此ItemBatcher欄位是 JSON 物件，指定在單一子工作流程執行中處理一組項目。在處理大型 CSV 檔案或 JSON 陣列或大型 Amazon S3 物件集時，請使用批次處理。

下列範例顯示ItemBatcher欄位的語法。在下列語法中，每個子工作流程執行應處理的最大項目數設定為 100。

```
{
  "ItemBatcher": {
    "MaxItemsPerBatch": 100
  }
```

```
}
```

根據預設，資料集中的每個項目都會做為輸入傳遞至個別子工作流程執行。例如，假設您將 JSON 檔案指定為包含下列陣列的輸入：

```
[
  {
    "verdict": "true",
    "statement_date": "6/11/2008",
    "statement_source": "speech"
  },
  {
    "verdict": "false",
    "statement_date": "6/7/2022",
    "statement_source": "television"
  },
  {
    "verdict": "true",
    "statement_date": "5/18/2016",
    "statement_source": "news"
  },
  ...
]
```

對於給定的輸入，每個子工作流程執行都會接收一個陣列項目作為其輸入。下列範例顯示子工作流程執行的輸入：

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

若要協助最佳化處理工作的效能和成本，請選取一個批次大小，以平衡項目數目與項目處理時間。如果您使用批次處理，「步驟函數」會將項目新增至項目陣列。然後，它會將陣列作為輸入傳遞至每個子工作流程執行。下列範例顯示一個批次，其中包含兩個項目作為輸入傳遞至子工作流程執行：

```
{
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
```

```
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    }
  ]
}
```

> **Tip**
>
> 若要進一步了解如何在工作流程中使用ItemBatcher欄位，請嘗試下列自學課程和研討會：
>
> - 在 Lambda 函數內處理整批資料
> - 在子工作流程執行內迭代批次中的項目
> - 單元 14 中的大規模並行化與分佈式地圖-工作坊的數據處理 AWS Step Functions

目錄

指定料號批次處理的欄位

若要批次處理項目，請指定要批次的最大項目數、最大批次大小或兩者。您必須為批次項目指定這些值之一。

每批次最大項目數

指定每個子工作流程執行處理的最大項目數。解釋器將數Items組中批處理的項目數限制為此值。如果同時指定批次編號和大小，解譯器會減少批次中的項目數，以避免超過指定的批次大小限制。

如果您未指定此值，但提供最大批次大小的值，則 Step Functions 會在每個子工作流程執行中處理盡可能多的項目，而不會超過以位元組為單位的最大批次大小。

例如，假設您使用包含 1130 個節點的輸入 JSON 檔案執行。如果您指定每個批次 100 的最大料號值，「步驟函數」會建立 12 個批次。其中 11 批包含 100 個項目，而第十二批包含剩餘的 30 個項目。

或者，您可以將每個批次的最大項目指定為「分散式對映」狀態輸入中現有鍵值組的參考路徑。此路徑必須解析為正整數。

例如，假設有下列輸入：

```
{
  "maxBatchItems": 500
}
```

您可以使用參照路徑指定要批次處理的最大項目數，如下所示：

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
    }
    ...
    ...
  }
}
```

> **Important**
>
> 您可以指定`MaxItemsPerBatch`或`MaxItemsPerBatchPath`子欄位，但不能同時指定兩者。

每批次最大 KB

指定批次的最大大小 (以位元組為單位)，最多 256 KB。如果同時指定最大批次編號和大小，Step Functions 會減少批次中的項目數，以避免超過指定的批次大小限制。

或者，您可以將最大批次大小指定為「分散式對映」狀態輸入中現有索引鍵值組的參考路徑。此路徑必須解析為正整數。

> **Note**
>
> 如果您使用批次處理，但未指定批次大小上限，則解譯器會在每個子工作流程執行中處理最多 256 KB 的項目。

例如，假設有下列輸入：

```
{
  "batchSize": 131072
}
```

您可以使用參照路徑指定最大批次大小，如下所示：

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
    }
    ...
    ...
  }
}
```

> **Important**
>
> 您可以指定MaxInputBytesPerBatch或MaxInputBytesPerBatchPath子欄位，但不能同時指定兩者。

批次輸入

或者，您也可以指定固定的 JSON 輸入，以包含在傳遞至每個子工作流程執行的每個批次中。Step Functions 會將此輸入與每個個別子工作流程執行的輸入合併。例如，給定下列項目陣列上事實檢查日期的固定輸入：

```
"ItemBatcher": {
```

```
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

每個子工作流程執行都會接收下列項目作為輸入

```
{
  "BatchInput": {
    "factCheck": "December 2022"
  },
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    },
    ...
  ]
}
```

ResultWriter

此`ResultWriter`欄位是 JSON 物件，用於指定 Amazon S3 位置，其中 Step Functions 會寫入由分散式地圖狀態開始的子工作流程執行結果。依預設，Step Functions 不會匯出這些結果。

> **Important**
>
> 確保用於匯出 Map Run 結果的 Amazon S3 儲存貯體與您的狀態機器處於相AWS 區域同AWS 帳戶的狀態機器下。否則，您的狀態機執行將失敗並顯示`States.ResultWriterFailed`錯誤。

如果您的輸出承載大小超過 256 KB，將結果匯出到 Amazon S3 儲存貯體會很有幫助。Step Functions 會合併所有子工作流程執行資料，例如執行輸入與輸出、ARN 以及執行狀態。然後，它會將具有

相同狀態的執行匯出到指定 Amazon S3 位置的個別檔案。下列範例顯示匯出子工作流程執行結果時ResultWriter欄位的語法。在此範例中，您會將結果儲存在名為的前置字元myOutputBucket內名稱的值區中csvProcessJobs。

```
{
  "ResultWriter": {
    "Resource": "arn:aws:states:::s3:putObject",
    "Parameters": {
      "Bucket": "myOutputBucket",
      "Prefix": "csvProcessJobs"
    }
  }
}
```

> **Tip**
>
> 在工作流程 Studio 中，您可以選取將地圖狀態結果匯出至 Amazon S3，以匯出子工作流程執行結果。然後，提供您要將結果匯出到的 Amazon S3 儲存貯體的名稱和前置詞。

Step Functions 需要適當的權限，才能存取您要匯出結果的值區和資料夾。如需所需 IAM 政策的相關資訊，請參閱的 IAM 政策 ResultWriter。

如果您匯出子工作流程執行結果，分散式地圖狀態執行會以下列格式傳回 Map Run ARN 和 Amazon S3 匯出位置的相關資料：

```
{
  "MapRunArn": "arn:aws:states:us-
east-2:123456789012:mapRun:csvProcess/Map:ad9b5f27-090b-3ac6-9beb-243cd77144a7",
  "ResultWriterDetails": {
    "Bucket": "myOutputBucket",
    "Key": "csvProcessJobs/ad9b5f27-090b-3ac6-9beb-243cd77144a7/manifest.json"
  }
}
```

Step Functions 將具有相同狀態的執行導出到各自的文件中。例如，如果子工作流程執行導致 500 次成功和 200 個失敗結果，則 Step Functions 會在指定的 Amazon S3 位置建立兩個檔案，以取得成功和失敗結果。在此範例中，成功結果檔案包含 500 個成功結果，而失敗結果檔案則包含 200 個失敗結果。

對於指定的執行嘗試，Step Functions 會根據您的執行輸出，在指定的 Amazon S3 位置建立下列檔案：

- `manifest.json`— 包含 Map Run 中繼資料，例如匯出位置、Map Run ARN 以及結果檔案的相關資訊。

  如果您redriven有 Map Run，則該`manifest.json`文件包含對 Map Run 的所有嘗試中所有成功的子工作流程執行的引用。但是，此檔案包含特定redrive失敗和擱置執行的參考。
- `SUCCEEDED_n.json`— 包含所有成功子工作流程執行的合併資料。 n 表示檔案的索引號碼。索引編號從 0 開始。例如 `SUCCEEDED_1.json`。
- `FAILED_n.json`— 包含所有失敗、逾時和中止子工作流程執行的合併資料。使用此檔案從失敗的執行中復原。 n 表示文件的索引。索引編號從 0 開始。例如 `FAILED_1.json`。
- `PENDING_n.json`— 包含因 Map Run 失敗或中止而未啟動的所有子工作流程執行的合併資料。 n 表示文件的索引。索引編號從 0 開始。例如 `PENDING_1.json`。

Step Functions 支援多達 5 GB 的個別結果檔案。如果檔案大小超過 5 GB，Step Functions 會建立另一個檔案來寫入剩餘的執行結果，並將索引號碼附加至檔案名稱。例如，如果`Succeeded_0.json`檔案的大小超過 5 GB，「Step Functions」會建立`Succeeded_1.json`檔案以記錄剩餘的結果。

如果您未指定匯出子工作流程執行結果，狀態機器執行會傳回子工作流程執行結果陣列，如下列範例所示：

> **Note**
>
> 如果傳回的輸出大小超過 256 KB，則狀態機器執行失敗並傳回`States.DataLimitExceeded`錯誤。

```
[
  {
    "statusCode": 200,
    "inputReceived": {
      "show_id": "s1",
      "release_year": "2020",
      "rating": "PG-13",
      "type": "Movie"
    }
  },
  {
```

```
    "statusCode": 200,
    "inputReceived": {
      "show_id": "s2",
      "release_year": "2021",
      "rating": "TV-MA",
      "type": "TV Show"
    }
  },
  ...
]
```

的 IAM 政策 ResultWriter

使用 Step Functions 主控台建立工作流程時，Step Functions 可以根據工作流程定義中的資源自動產生 IAM 政策。這些原則包括允許狀態機器角色呼叫分散式對應狀態的 StartExecution API 動作所需的最低權限。這些政策還包括存取AWS資源所需的最低權限 Step Functions，例如 Amazon S3 儲存貯體和物件以及 Lambda 函數。我們強烈建議您僅在 IAM 政策中加入必要的許可。例如，如果您的工作流程包含分散式模式的Map狀態，請將您的政策範圍縮減到包含資料集的特定 Amazon S3 儲存貯體和資料夾。

> **Important**
>
> 如果您指定 Amazon S3 儲存貯體和物件或前置詞，其中包含分散式地圖狀態輸入中現有鍵值組的參考路徑，請確定您已更新工作流程的 IAM 政策。將政策範圍縮小到值區和路徑在執行時期解析為的物件名稱。

下列 IAM 政策範例授予使用 PutObject API 動作將子工作流程執行結果寫入 Amazon S3 儲存貯體中名為 *CSVJobs* 的資料夾所需的最低權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

如果您要寫入子工作流程執行結果的 Amazon S3 儲存貯體使用AWS Key Management Service(AWS KMS)金鑰加密，您必須在 IAM 政策中包含必要的AWS KMS許可。如需詳細資訊，請參閱 AWS KMS key 加密 Amazon S3 儲存貯體的 IAM 許可。

剖析輸入 CSV 檔案

因為沒有標準化的格式來建立和維護 CSV 檔案中的資料，因此「Step Functions 數」會根據下列規則剖析 CSV 檔案：

- 逗號 (,) 是用來分隔個別欄位的分隔符號。
- 換行符是分隔各個記錄的分隔符。
- 欄位會被視為字串。若要進行資料類型轉換，請使用States.StringToJson內在函數ItemSelector。
- 包含字串時，不需要雙引號 (」「)。但是，以雙引號括住的字串可以包含逗號和換行符，而不會用作分隔符號。
- 通過重複它們來逃避雙引號。
- 如果資料列中的欄位數目小於標頭中的欄位數目，Step Functions 會為遺漏值提供空字串。
- 如果列中的欄位數目大於標題中的欄位數目，則「Step Functions 數」會略過其他欄位。

Example 的剖析輸入 CSV 檔案

假設您提供了一個名為的 CSV 文件*myCSVInput.csv*包含一行作為輸入。然後，您已將此檔案儲存在 Amazon S3 儲存貯體中*my-bucket*。CSV 檔案如下所示。

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

以下狀態機讀取此 CSV 文件並使用ItemSelector轉換某些欄位的資料類型。

```
{
  "StartAt": "Map",
  "States": {
```

```
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "Pass",
        "States": {
          "Pass": {
            "Type": "Pass",
            "End": true
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemReader": {
        "Resource": "arn:aws:states:::s3:getObject",
        "ReaderConfig": {
          "InputType": "CSV",
          "CSVHeaderLocation": "GIVEN",
          "CSVHeaders": [
            "MyLetters",
            "MyNumbers",
            "MyString",
            "MyObject",
            "MyArray"
          ]
        },
        "Parameters": {
          "Bucket": "my-bucket",
          "Key": "myCSVInput.csv"
        }
      },
      "ItemSelector": {
        "MyLetters.$": "$$.Map.Item.Value.MyLetters",
        "MyNumbers.$": "States.StringToJson($$.Map.Item.Value.MyNumbers)",
        "MyString.$": "$$.Map.Item.Value.MyString",
        "MyObject.$": "States.StringToJson($$.Map.Item.Value.MyObject)",
        "MyArray.$": "States.StringToJson($$.Map.Item.Value.MyArray)"
      }
    }
```

```
  }
}
```

當你運行這個狀態機，它產生以下輸出。

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

內容物件

上下文對象是一個內部 JSON 結構，可在執行期間使用，並包含有關狀態機器和執行的信息。這可讓您的工作流程存取關於其特定執行的資訊。您可以從下列欄位存取前後關聯物件：

- `InputPath`
- `OutputPath`
- `ItemsPath`(在「地圖」狀態中)
- `Variable`(在「選擇」狀態中)
- `ResultSelector`
- `Parameters`
- 變量到變量比較運算符

內容物件格式

內容物件包含與狀態機器，狀態、執行和任務相關的資訊。這個 JSON 物件包含每種資料類型的節點，而且採用下列格式：

```
{
    "Execution": {
        "Id": "String",
        "Input": {},
        "Name": "String",
        "RoleArn": "String",
        "StartTime": "Format: ISO 8601",
        "RedriveCount": Number,
        "RedriveTime": "Format: ISO 8601"
    },
    "State": {
        "EnteredTime": "Format: ISO 8601",
        "Name": "String",
        "RetryCount": Number
    },
    "StateMachine": {
        "Id": "String",
        "Name": "String"
    },
    "Task": {
        "Token": "String"
    }
}
```

執行期間，內容物件所存取 Parameters 欄位將會填入內容物件及其相關資料。如果 Parameters 欄位的值不在任務狀態範圍中，Task 欄位就會為空值。

如果您尚redriven未執行，則RedriveCount上下文對象的值為 0。此外，上RedriveTime下文對象只有在您執行時才可redriven用。如果您有 redriven a Map Run，RedriveTime前後關聯物件僅適用於類型為「標準」的子工作流程。對於具有 Express 類型的子工作流程的redriven地圖運行，RedriveTime不可用。

正在進行中執行的內容包含以下格式的規格。

```
{
    "Execution": {
        "Id": "arn:aws:states:us-
east-1:123456789012:execution:stateMachineName:executionName",
        "Input": {
           "key": "value"
        },
        "Name": "executionName",
```

```
        "RoleArn": "arn:aws:iam::123456789012:role...",
        "StartTime": "2019-03-26T20:14:13.192Z"
    },
    "State": {
        "EnteredTime": "2019-03-26T20:14:13.192Z",
        "Name": "Test",
        "RetryCount": 3
    },
    "StateMachine": {
        "Id": "arn:aws:states:us-east-1:123456789012:stateMachine:stateMachineName",
        "Name": "stateMachineName"
    },
    "Task": {
        "Token": "h7XRiCdLtd/83p1E0dMccoxlzFhglsdkzpK9mBVKZsp7d9yrT1W"
    }
}
```

> **Note**
>
> 如需 Map 狀態相關的內容物件資料，請參閱Map 狀態的內容物件資料。

存取內容物件

若要存取內容物件，請先將 `.$` 附加到結尾來指定參數名稱，就像您利用路徑來選擇狀態輸入的方式。接下來，如要存取內容物件資料而不要存取輸入，請在路徑名稱之前加上 `$$.`。這告訴 AWS Step Functions 使用路徑來選擇上下文對象的節點。

下列範例顯示如何存取前後關聯物件，例如執行 ID、名稱、開始時間和redrive計數。

- 擷取執行 ARN 並將其傳遞至下游服務
- 在「通過」狀態下存取執行開始時間和名稱
- 訪問執行的redrive計數

擷取執行 ARN 並將其傳遞至下游服務

此範例任務狀態使用路徑擷取執行 Amazon 資源名稱 (ARN)，並將其傳遞至 Amazon Simple Queue Service (Amazon SQS) 訊息。

```
{
```

```
  "Order Flight Ticket Queue": {
    "Type": "Task",
    "Resource": "arn:aws:states:::sqs:sendMessage",
    "Parameters": {
      "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/flight-purchase",
      "MessageBody": {
        "From": "YVR",
        "To": "SEA",
        "Execution.$": "$$.Execution.Id"
      }
    },
    "Next": "NEXT_STATE"
  }
}
```

如需有關在呼叫整合服務時使用任務字符的詳細資訊，請參閱[等候傳回任務字符的回呼](#)。

在「通過」狀態下存取執行開始時間和名稱

```
{
  "Comment": "Accessing context object in a state machine",
  "StartAt": "Get execution context data",
  "States": {
    "Get execution context data": {
      "Type": "Pass",
      "Parameters": {
        "startTime.$": "$$.Execution.StartTime",
        "execName.$": "$$.Execution.Name"
      },
      "ResultPath": "$.executionContext",
      "End": true
    }
  }
}
```

訪問執行的redrive計數

下列 Task 狀態定義範例顯示如何存取執行[redrive](#)計數。

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke",
  "OutputPath": "$.Payload",
```

```
  "Parameters": {
    "Payload": {
      "Number.$": "$$.Execution.RedriveCount"
    },
    "FunctionName": "arn:aws:lambda:us-east-2:123456789012:function:functionName"
  },
  "End": true
}
```

Map 狀態的內容物件資料

處理 Map 狀態時，內容物件中有兩個額外的可用項目：Index 和 Value。針對每個Map狀態反覆運算，Index包含目前正在處理之陣列項目的索引編號，同時Value包含正在處理的陣列項目。在某個Map狀態中，前後關聯物件包含下列資料：

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

這些只能在某個Map狀態下使用，並且可以在ItemSelector欄位中指定。

> **Note**
>
> 您必須在主要 Map 狀態的 ItemSelector 區塊中定義內容物件的參數，而不是在 ItemProcessor 區段包含的狀態內。

假設狀態機器具有簡單 Map 狀態，我們可以從內容物件注入資訊，如下所示。

```
{
  "StartAt": "ExampleMapState",
  "States": {
    "ExampleMapState": {
      "Type": "Map",
      "ItemSelector": {
        "ContextIndex.$": "$$.Map.Item.Index",
        "ContextValue.$": "$$.Map.Item.Value"
      },
```

```
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "INLINE"
        },
        "StartAt": "TestPass",
        "States": {
          "TestPass": {
            "Type": "Pass",
            "End": true
          }
        }
      },
      "End": true
    }
  }
}
```

如果您使用下列輸入執行先前的狀態機器，Index 和 Value 會插入到輸出中。

```
[
  {
    "who": "bob"
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
]
```

執行的輸出會傳回三個反覆Value項目的值Index和項目，如下所示：

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
```

```
      "who": "meg"
    }
  },
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

數據流模擬器

您可以在 Step Functions 主控台中設計、實作和偵錯工作流程。您也可以透過JsonPath輸入和輸出資料處理來控制工作流程中的資料流程。使用資料流程模擬器，您可以模擬工作流程程序資料在執行階段中任務狀態的順序。使用模擬器，您可以了解如何過濾和操作數據，因為它從一個狀態流到另一個狀態。它會模擬步驟函式用來處理和控制 JSON 資料流程的下列每個欄位：

InputPath

選取整個輸入有效負載的 WHERE 部分，以用作工作的輸入。如果您指定此欄位，「步驟函數」會先套用此欄位。

##

指定調用任務之前輸入的外觀。透過此Parameters欄位，您可以建立索引鍵值配對的集合，這些索引鍵值配對會當做輸入傳遞至AWS 服務整合，例如AWS Lambda函數。這些值可以是靜態的，也可以是從狀態輸入或工作流程前後關聯物件動態選取的。

ResultSelector

決定要從工作輸出中選擇的項目。使用該ResultSelector字段，您可以創建鍵值對的集合，以替換狀態的結果並將ResultPath該集合傳遞給。

ResultPath

決定要放置工作輸出的位置。使用ResultPath來判斷狀態的輸出是其輸入的副本、產生的結果，還是兩者的組合。

OutputPath

決定要傳送至下一個狀態的內容。使用OutputPath，您可以過濾掉不需要的信息，並僅傳遞您關心的 JSON 數據部分。

在本主題中

使用資料流模擬器

模擬器會在您套用輸入和輸出資料處理欄位之前和之後，提供即時的資料side-by-side比較。若要使用模擬器，請指定 JSON 輸入。然後，通過每個輸入和輸出處理字段對其進行評估。模擬器會自動驗證您的 JSON 輸入並反白顯示任何語法錯誤。

若要使用資料流程模擬器

在下列步驟中，您會提供 JSON 輸入並套用`InputPath`和`##`欄位。您也可以套用其他可用欄位並檢視其輸出。

1. 開啟步驟功能主控台。
2. 在導覽窗格中，選擇 [資料流程模擬器]。
3. 在 [狀態] 輸入區域中，以下列 JSON 資料取代預先填入的範例 JSON 資料。然後選擇 Next (下一步)。

```
{
  "data": {
    "firstname": "Jane",
    "lastname": "Doe",
    "identity": {
      "email": "jdoe@example.com",
      "ssn": "123-45-6789"
    },
    "address": {
      "street": "123 Main St",
      "city": "Columbus",
      "state": "OH",
      "zip": "43219"
    }
  }
}
```

4. 對於 InputPath，輸入**`$.data.address`**以選取輸入 JSON 資料的位址節點。

   「狀態輸入後 InputPath」方塊會顯示下列結果。

```
{
  "street": "123 Main St",
  "city": "Columbus",
  "state": "OH",
  "zip": "43219"
}
```

5. 選擇 下一步。
6. 套用Parameters欄位以將結果 JSON 資料轉換為字串。若要轉換資料，請執行下列動作：
   - 在「參數」方塊中，輸入下列程式碼以建立名為的字串addressString。

     ```
     {
         "addressString.$": "States.Format('{}. {}, {} - {}', $.street, $.city,
      $.state, $.zip)"
       }
     ```
7. 在「參數之後篩選的輸入」方塊中，檢視Parameters欄位應用程式的結果。

使用資料流程模擬器的注意事項

在使用資料流程模擬器之前，請考慮其限制，包括但不限於：

- 不支援的篩選器

  模擬器中的篩選器運算式與 Step Functions 服務中的行為不同。這包括使用下列語法的運算式：[?(expression)]。以下是不支援的運算式清單。如果使用，這些運算式在評估後可能不會傳回預期的結果。
  - $..book[?(@.isInStock==true)]
  - $..book[?(@.price > 10.0)].title
- 單一項目陣列的JsonPath評估不正確

  如果您使用返回單個數組項目的JsonPath表達式過濾數據，則模擬器返回沒有數組的項目。例如，考慮下面的數據數組，稱為items：

```
{
  "items": [
    {
```

```
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

鑑於此`items`數組，如果您**`$..comment`**在InputPath字段中輸入，則需要以下輸出：

```
[
  "nice shoe"
]
```

但是，數據流模擬器返回以下輸出：

```
"nice shoe"
```

對於包含多個項目的數組的JsonPath評估，模擬器返回預期的輸出。

使用版本和別名管理持續部署

您可以使用 Step Functions 透過狀態機版本和別名來管理工作流程的連續部署。版本是您可以執行的狀態機器的編號、不可變快照集。別名是最多兩個版本的狀態機的指針。

您可以維護多個版本的狀態機器，並在生產工作流程中管理它們的部署。使用別名，您可以在不同的工作流程版本之間路由流量，並逐步將這些工作流程部署到生產環境。

此外，您可以使用版本或別名來啟動狀態機器執行。如果您在啟動狀態機器執行時未使用版本或別名，Step Functions 會使用狀態機器定義的最新修訂版本。

狀態機修訂版

狀態機可以有一個或多個修訂版。當您使用 UpdateStateMachineAPI 動作更新狀態機器時，它會建立新的狀態機器修訂版本。修訂版是狀態機器定義和組態的不可變唯讀快照。您無法從修訂版啟動狀態機執行，並且修訂版本沒有 ARN。修訂版本具有一個`revisionId`，它是一個通用唯一標識符（UUID）。

目錄

狀態機版本

版本是狀態機器的編號、不可變的快照集。您可以從對該狀態機器進行的最新修訂版本中發佈版本。每個版本都有一個唯一的 Amazon 資源名稱（ARN）。此 ARN 是狀態機 ARN 和版本號以冒號 (:) 分隔的組合。下列範例顯示狀態機器版本 ARN 的格式。

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

若要開始使用狀態機版本，您必須發佈第一個版本。發佈版本後，您可以使用 ARN 版本叫用 StartExecutionAPI 動作。您無法編輯版本，但可以更新狀態機器並發佈新版本。您也可以發佈多個版本的狀態機器。

當您發佈狀態機器的新版本時，Step Functions 會為其指派版本號碼。版本號碼從 1 開始，每個新版本都會單調增加。版本號不會重複用於給定的狀態機器。如果您刪除狀態機器的版本 10，然後發佈新版本，Step Functions 會將其發佈為版本 11。

對於狀態機的所有版本，以下屬性都相同：

- 狀態機的所有版本都共用相同類型 (標準或快速)。
- 您無法在版本之間變更狀態機器的名稱或建立日期。
- 標籤會全域套用至狀態機器。您可以使用TagResource和 UntagResourceAPI 動作管理狀態機器的標籤。

狀態機器也包含屬於每個版本和的屬性revision，但這些屬性在兩個指定版本或修訂之間可能會有所不同。這些內容包括狀態機器定義、IAM 角色、追蹤組態和記錄組態。

目錄

發佈狀態機器版本 (主控台)

您最多可以發佈 1000 個版本的狀態機器。若要要求提高此軟限制，請使用中的「Sup port 中心」頁面AWS Management Console。您可以從控制台手動刪除未使用的版本，也可以通過調用DeleteStateMachineVersionAPI 操作來手動刪除。

發佈狀態機器版本的步驟

1. 開啟 Step Functions 主控台，然後選擇現有的狀態機器。
2. 在狀態機器詳細資料頁面上，選擇編輯。
3. 視需要編輯狀態機定義，然後選擇 [儲存]。
4. 選擇 Publish version (發佈版本)。
5. (選擇性) 在顯示之對話方塊的「描述」欄位中，輸入有關狀態機器版本的簡短說明。
6. 選擇 Publish (發佈)。

> **Note**
>
> 當您發佈狀態機器的新版本時，Step Functions 會為其指派版本號碼。版本號碼從 1 開始，每個新版本都會單調增加。版本號不會重複用於給定的狀態機器。如果您刪除狀態機器的版本 10，然後發佈新版本，Step Functions 會將其發佈為版本 11。

使用 Step Functions API 作業管理版本

Step Functions 提供下列 API 作業來發佈和管理狀態機版本：

- PublishStateMachineVersion— 從狀態機器revision的目前發佈版本。
- UpdateStateMachine— 如果您更新狀態機器並在相同請求`true`中將`publish`參數設定為，則會發佈新的狀態機版本。
- CreateStateMachine— 如果將`publish`參數設定為，則發佈狀態機器的第一個修訂版本`true`。
- ListStateMachineVersions— 列出指定狀態機 ARN 的版本。
- DescribeStateMachine— 傳回中指定之版本 ARN 的狀態機器版本詳細資訊。`stateMachineArn`
- DeleteStateMachineVersion— 刪除狀態機器版本。

若要從名為的狀態機器的目前版本中發佈新版本 AWS Command Line Interface，請*`myStateMachine`*使用以下`publish-state-machine-version`指令：

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

回應會傳回 `stateMachineVersionArn`。例如，上一個指令會傳回的回應`arn:aws:states:us-east-1:`*`123456789012`*`:stateMachine:`*`myStateMachine`*`:1`。

> **Note**
>
> 當您發佈狀態機器的新版本時，Step Functions 會為其指派版本號碼。版本號碼從 1 開始，每個新版本都會單調增加。版本號不會重複用於給定的狀態機器。如果您刪除狀態機器的版本 10，然後發佈新版本，Step Functions 會將其發佈為版本 11。

從主控台執行狀態機器版本

若要開始使用狀態機版本，您必須先從目前狀態機器發佈版本revision。若要發佈版本，請使用「Step Functions」主控台或叫用 PublishStateMachineVersionAPI 動作。您也可以使用名為的選用參數叫用 UpdateStateMachineAliasAPI 動作，publish以更新狀態機器並發佈其版本。

您可以使用主控台或叫用 StartExecutionAPI 動作並提供版本 ARN 來啟動版本的執行。您也可以使用別名來啟動版本的執行。根據其路由組態，別名會將流量路由至特定版本。

如果您在不使用版本的情況下啟動狀態機器執行，則 Step Functions 會使用狀態機器的最新修訂版本來執行。如需有關「Step Functions 式」如何將執行與版本相關聯的資訊，請參閱將執行與版本或別名相關聯。

若要使用狀態機版本開始執行

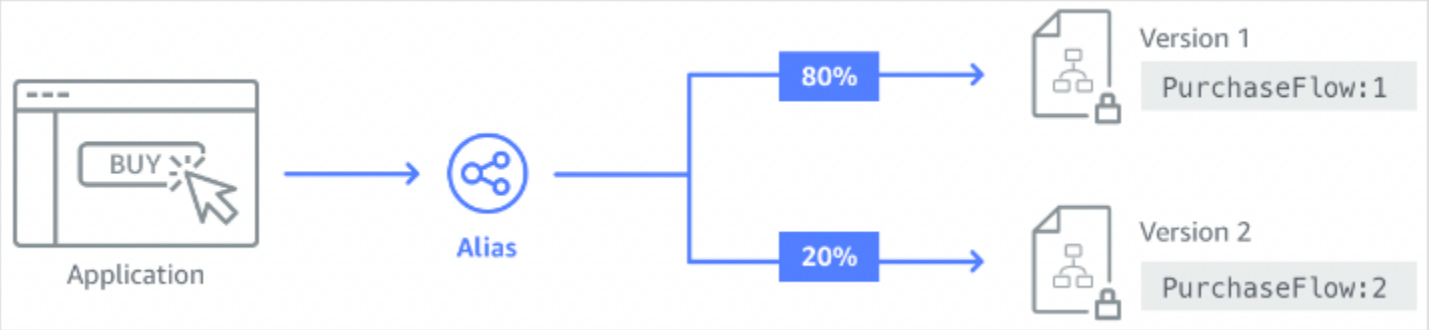
1. 開啟 Step Functions 主控台，然後選擇您已針對其發佈一或多個版本的現有狀態機器。若要瞭解如何發佈版本，請參閱發佈狀態機器版本 (主控台)。
2. 在狀態機器詳細資料頁面上，選擇版本索引標籤。
3. 在「版本」區段中，執行下列操作：
   a. 選取您要開始執行的版本。
   b. 選擇 Start execution (開始執行)。
4. (選擇性) 在 [開始執行] 對話方塊中，輸入執行項目的名稱。
5. (選擇性)，輸入執行輸入，然後選擇 [開始執行]。

狀態機別名

別名是最多兩個版本的相同狀態機的指針。您可以為狀態機器建立多個別名。每個別名都有一個唯一的 Amazon 資源名稱（ARN）。別名 ARN 是狀態機器 ARN 和別名名稱的組合，以冒號 (:) 分隔。下列範例顯示狀態機別名 ARN 的格式。

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

您可以使用別名在兩個狀態機器版本之一之間路由流量。您也可以建立指向單一版本的別名。別名只能指向狀態機器版本。您不能使用別名來指向另一個別名。您也可以更新別名，以指向不同版本的狀態機器。



目錄

建立狀態機器別名 (主控台)

您可以使用 Step Functions 主控台或叫用 CreateStateMachineAliasAPI 動作，為每個狀態機器建立最多 100 個別名。若要要求提高此軟限制，請使用中的「Sup port 中心」頁面AWS Management Console。從主控台或叫用 DeleteStateMachineAliasAPI 動作來刪除未使用的別名。

建立狀態機別名的步驟

1. 開啟 Step Functions 主控台，然後選擇現有的狀態機器。
2. 在狀態機器詳細資訊頁面上，選擇別名索引標籤。
3. 選擇 [建立新別名]。
4. 在 Create alias (建立別名) 頁面，執行下列動作：
   a. 輸入別名。
   b. (選用) 輸入別名的 Description (描述)。
5. 若要在別名上設定路由，請參閱別名路由組態。
6. 選擇 [建立別名]。

使用 Step Functions API 作業管理別名

Step Functions 提供下列 API 作業，可用來建立和管理狀態機器別名或取得別名相關資訊：

- CreateStateMachineAlias— 為狀態機器建立別名。
- DescribeStateMachineAlias— 傳回有關狀態機器別名的詳細資訊。
- ListStateMachineAliases— 列出指定狀態機 ARN 的別名。
- UpdateStateMachineAlias-透過修改現有狀態機器別名的description或來更新其組態routingConfiguration。
- DeleteStateMachineAlias— 刪除狀態機器版本。

若要建立指向使*PROD*用命名之狀態機器版本 1 的別名 AWS Command Line Interface，請*myStateMachine*使用create-state-machine-alias指令。

```
aws stepfunctions create-state-machine-alias --name PROD  --routing-
configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

別名路由組態

您可以使用別名在兩個版本的狀態機器之間路由執行流量。例如，假設您想要啟動狀態機的新版本。您可以在別名上設定路由，以降低部署新版本所涉及的風險。透過設定路由，您可以將大部分流量傳送到狀態機器的早期測試版本。然後，新版本可以獲得較小的百分比，直到您可以確認可以安全地向前滾動新版本。

若要定義路由組態，請確定您已發佈別名指向的兩個狀態機器版本。當您從別名開始執行時，Step Functions 會隨機選擇要從路由組態中指定的版本執行的狀態機器版本。它會根據您指派給別名路由組態中每個版本的流量百分比為基礎。

若要在別名上設定路由組態

- 在 [建立別名] 頁面的 [路由組態] 下，執行下列動作：
  a. 對於版本，選擇別名指向的第一個狀態機器版本。
  b. 選取「在兩個版本之間分割流量」核取方塊。

  > **Tip**
  > 若要指向單一版本，請清除「在兩個版本之間分割流量」核取方塊。

  c. 在版本中，選擇別名必須指向的第二個版本。

d. 在「流量百分比」欄位中，指定要路由到每個版本的流量百分比。例如，輸入**60**和**40**將 60% 的執行流量路由到第一個版本，將 40% 的流量路由傳送至第二個版本。

   合併的流量百分比必須等於 100%。

使用別名運行狀態機（控制台）

您可以從控制台或使用別名的 ARN 調用 StartExecutionAPI 動作，以使用別名來啟動狀態機器執行。Step Functions 然後運行別名指定的版本。根據預設，如果您在啟動狀態機器執行時未指定版本或別名，Step Functions 會使用最新的修訂版本。

若要使用別名啟動狀態機執行

1. 開啟 Step Functions 主控台，然後選擇您已為其建立別名的現有狀態機器。如需建立別名的資訊，請參閱建立狀態機器別名 (主控台)。
2. 在狀態機器詳細資訊頁面上，選擇別名索引標籤。
3. 在「別名」段落中，執行下列動作：
   a. 選取您要開始執行的別名。
   b. 選擇 Start execution (開始執行)。
4. (選擇性) 在 [開始執行] 對話方塊中，輸入執行項目的名稱。
5. 如果需要，請輸入執行輸入，然後選擇 [開始執行]。

版本和別名的授權

若要使用版本或別名叫用步驟函數 API 動作，您需要適當的權限。若要授權版本或別名來叫用 API 動作，步驟函式會使用狀態機器的 ARN，而不是使用版本 ARN 或別名 ARN。您也可以縮小特定版本或別名的權限範圍。如需詳細資訊，請參閱設定權限的範圍。

您可以使用名為的狀態機器的下列 IAM 政策範例*myStateMachine*來叫用 CreateStateMachineAliasAPI 動作來建立狀態機器別名。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "states:CreateStateMachineAlias",
```

```
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine"
    }
  ]
}
```

當您設定權限以允許或拒絕使用狀態機版本或別名存取 API 動作時，請考慮下列事項：

- 如果您使用CreateStateMachine和 UpdateStateMachineAPI 動作的publish參數發佈新的狀態機器版本，您還需要 PublishStateMachineVersionAPI 動作的ALLOW權限。
- DeleteStateMachineAPI 動作會刪除與狀態機器相關聯的所有版本和別名。

在本主題中

設定版本或別名的權限範圍

您可以使用限定詞進一步縮小版本或別名所需的授權權限範圍。限定詞是指版本號碼或別名名稱。您可以使用限定詞來限定狀態機器。下列範例是使用名稱PROD為限定詞的別名的狀態機器 ARN。

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

如需有關合格和不合格 ARN 的詳細資訊，請參閱。將執行與版本或別名相關聯

您可以使用 IAM 政策Condition陳述式states:StateMachineQualifier中指定的選用內容金鑰來縮小許可範圍。例如，名為的狀態機器的下列 IAM 政策會myStateMachine拒絕存取以叫用名為PROD或版本1的別名叫用 DescribeStateMachineAPI 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "states:DescribeStateMachine",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "states:StateMachineQualifier": [
            "PROD",
            "1"
```

```
          ]
        }
      }
    }
  ]
}
```

下列清單會指定 API 動作，您可以使用`StateMachineQualifier`內容金鑰設定權限範圍。

- CreateStateMachineAlias
- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

將狀態機器執行與版本或別名相關聯

Step Functions 會根據您用來叫用 StartExecutionAPI 動作的 Amazon 資源名稱 (ARN)，將執行與版本或別名相關聯。Step Functions 在執行開始時執行此操作。

您可以使用限定或不合格的 ARN 啟動狀態機執行。

- 合格 ARN — 指以版本號或別名後綴的狀態機器 ARN。

  下列合格 ARN 範例是指名為`3myStateMachine`的狀態機器版本。

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
  ```

  下列合格 ARN 範例是指名為`PROD`的狀態機器名為的別名`myStateMachine`。

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
  ```

- 不合格 ARN — 指沒有版本號或別名後綴的狀態機器 ARN。

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

例如，如果您的合格 ARN 引用版本3，則 Step Functions 將執行與此版本相關聯。它不會將執行與任何指向版本的別名相關聯3。

如果您的合格 ARN 引用別名，則 Step Functions 將執行與該別名和別名指向的版本相關聯。一個執行只能與一個別名相關聯。

> **Note**
>
> 如果您使用不合格的 ARN 開始執行，即使版本使用相同的狀態機器，Step Functions 也不會將該執行與版本相關聯。revision例如，如果版本 3 使用最新的修訂版，但您以不合格的 ARN 開始執行，則 Step Functions 不會將該執行與版本 3 產生關聯。

在本主題中

檢視以版本或別名開始的執行項目

Step Functions 提供了以下方法，您可以在其中查看以版本或別名開始的執行：

使用 API 動作

您可以叫用DescribeExecution和 ListExecutionsAPI 動作來檢視與版本或別名相關聯的所有執行。這些 API 動作會傳回用來啟動執行之版本或別名的 ARN。這些動作也會傳回其他詳細資訊，包括執行的狀態和 ARN。

您也可以提供狀態機器別名 ARN 或版本 ARN，以列出與特定別名或版本相關聯的執行。

下列 ListExecutionsAPI 動作的範例回應會顯示用來啟動名為*myFirstExecution*的狀態機器執行之別名的 ARN。

下列程式碼片段中的*##*文字代表資源特定的資訊。

```
{
```

```
    "executions": [
        {
            "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:myStateMachine:myFirstExecution",
            "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine",
            "stateMachineAliasArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:PROD",
            "name": "myFirstExecution",
            "status": "SUCCEEDED",
            "startDate": "2023-04-20T23:07:09.477000+00:00",
            "stopDate": "2023-04-20T23:07:09.732000+00:00"
        }
    ]
}
```

使用 Step Functions 主控台

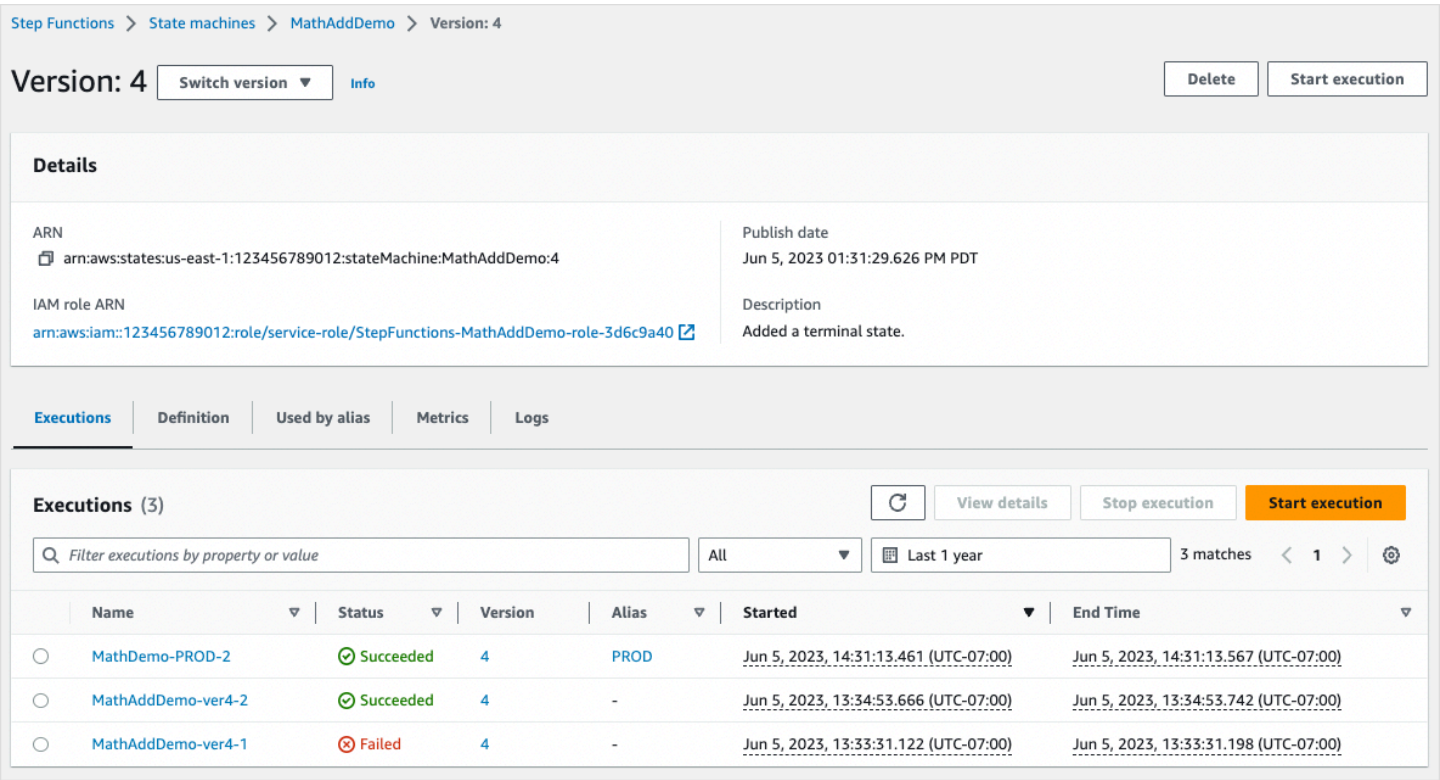
您也可以從 Step Functions 主控台檢視由版本或別名啟動的執行。下列程序顯示如何檢視以特定版本開始的執行項目：

1. 開啟 Step Functions 主控台，然後選擇您已針對其發佈版本或建立別名的現有狀態機器。此範例顯示如何檢視以特定狀態機器版本開始的執行。
2. 選擇 [版本] 索引標籤，然後從 [版本] 清單中選擇版本。

   > **Tip**
   >
   > 依內容或值方塊篩選，以搜尋特定版本。

3. 在 [版本詳細資料] 頁面上，您可以看到以所選版本開始的所有進行中和過去狀態機器執行的清單。

下圖顯示 [版本詳細資料] 主控台頁面。此頁面列出了名為*MathAddDemo*的狀態機器的第 4 版啟動的執行。此清單也會顯示由名稱為的別名啟動的執行項目*PROD*。此別名將執行流量路由至第 4 版。



使用 CloudWatch 指標

對於您以 a 開始的每個狀態機器執行Qualified ARN，Step Functions 會發出與目前發出的指標相同名稱和值的其他度量。這些其他量度包含您開始執行時所使用的每個版本識別碼和別名名稱的維度。使用這些指標，您可以在版本層級監視狀態機器執行，並判斷何時可能需要復原案例。您也可以根據這些指標建立 Amazon CloudWatch 警示。

Step Functions 會針對您以別名或版本開始的執行發出下列量度：

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`
- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

如果您使用 ARN 版本開始執行，則「Step Functions」會發佈具有`StateMachineArn`和維度的第二個`StateMachineArn`量`Version`度。

如果您使用別名 ARN 開始執行，Step Functions 會發出下列指標：

- 不合格 ARN 和版本的兩個指標。
- 具有StateMachineArn和維度的量Alias度。

別名和版本部署範例

下列 Canary 部署技術範例顯示如何使用AWS Command Line Interface. 在此範例中，您建立的別名會將 20% 的執行流量路由至新版本。然後，它將剩餘的 80% 路由到早期版本。若要部署新的狀態機器版本並使用別名轉移執行流量，請完成以下步驟：

1. 從目前的狀態機器修訂發佈版本。

   使用中的publish-state-machine-versionAWS CLI指令從名為的狀態機器的目前版本發佈版本 *myStateMachine*：

   ```
   aws stepfunctions publish-state-machine-version --state-machine-arn
    arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
   ```

   回應會stateMachineVersionArn傳回您發佈的版本。例如：arn:aws:states:us-east-1:*123456789012*:stateMachine:*myStateMachine*:1。
2. 建立指向狀態機版本的別名。

   使用指create-state-machine-alias令建立指向以下版本 1 *PROD* 的名稱的別名*myStateMachine*：

   ```
   aws stepfunctions create-state-machine-alias --name PROD --routing-
   configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
   ```
3. 驗證別名啟動的執行使用正確的發佈版本。

   透過**PROD**在start-execution命令中*myStateMachine*提供別名的 ARN 來啟動的新執行：

   ```
   aws stepfunctions start-execution
     --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachineAlias:myStateMachine:PROD
     --input "{}"
   ```

   如果您在StartExecution請求中提供狀態機 ARN，它會使用最新revision的狀態機器，而不是別名中指定的版本來開始執行。
4. 更新狀態機定義並發佈新版本。

更新*myStateMachine*並發布其新版本。要做到這一點，使用update-state-machine命令的可選publish參數：

```
aws stepfunctions update-state-machine
    --state-machine-arn arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine
    --definition $UPDATED_STATE_MACHINE_DEFINITION
    --publish
```

回應會stateMachineVersionArn傳回新版本的。例如：arn:aws:states:us-east-1:*123456789012*:stateMachine:*myStateMachine*:2。

5. 更新別名以指向兩個版本，並設置別名的路由配置。

   使用update-state-machine-alias指令更新別名的路由組態PROD。設定別名，讓 80% 的執行流量進入第 1 版，剩餘的 20% 會移至第 2 版：

   ```
   aws stepfunctions update-state-machine-alias --state-machine-alias-arn
    arn:aws:states:us-east-1:123456789012:stateMachineAlias:myStateMachine:PROD
    --routing-configuration "[{\"stateMachineVersionArn\":
   \"arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1\",
   \"weight\":80}, {\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:2\",\"weight\":20}]"
   ```

6. 將版本 1 取代為版本 2。

   確認新的狀態機器版本正常運作之後，您可以部署新的狀態機器版本。若要這麼做，請再次更新別名，將 100% 的執行流量指派給新版本。

   使用update-state-machine-alias指令將版本 2 的別PROD名路由配置設定為 100%：

   ```
   aws stepfunctions update-state-machine-alias --state-machine-alias-arn
    arn:aws:states:us-east-1:123456789012:stateMachineAlias:myStateMachine:PROD
    --routing-configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:2\",\"weight\":100}]"
   ```

> **Tip**
>
> 若要復原第 2 版的部署，請編輯別名的路由組態，將 100% 的流量轉移到新部署的版本。

```
aws stepfunctions update-state-machine-alias
  --state-machine-alias-arn arn:aws:states:us-
east-1:123456789012:stateMachineAlias:myStateMachine:PROD
  --routing-configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

您可以使用版本和別名來執行其他類型的部署。例如，您可以執行新版本的狀態機器的滾動部署。若要這麼做，請逐漸增加指向新版本之別名路由組態中的加權百分比。

您也可以使用版本和別名來執行藍/綠部署。為此，請創建一個名為的別名，green該別名運行狀態機的當前版本 1。然後，建立另一個名blue為執行新版本的別名，例如2。若要測試新版本，請將執行流量傳送至blue別名。當您確定新版本正常運作時，請更新green別名以指向新版本。

執行狀態機器版本的逐步部署

滾動式部署是一種部署策略，會慢慢地以新版應用程式取代舊版應用程式。若要執行狀態機器版本的滾動式部署，請逐步傳送越來越多的執行流量到新版本。流量量和增加速率是您設定的參數。

您可以使用下列其中一個選項來執行版本的滾動式部署：

- 步驟功能控制台— 建立指向相同狀態機器的兩個版本的別名。對於此別名，您可以設定路由組態以在兩個版本之間轉移流量。如需使用主控台推出版本的詳細資訊，請參閱版本和Aliases。
- 用於指令碼AWS CLI和開發套件— 使用建立殼層指令碼AWS CLI或AWSSDK。如需詳細資訊，請參閱下列使用章節AWS CLI和AWSSDK。
- AWS CloudFormation模板— 使用AWS::StepFunctions::StateMachineVersion和AWS::StepFunctions::StateMachineAlia於發佈多個狀態機器版本並建立別名以指向其中一個或兩個版本的資源。

使用AWS CLI部署新的狀態機版本

本節中的範例指令碼顯示如何使用AWS CLI逐漸將流量從以前的狀態機版本轉移到新的狀態機版本。您可以使用此示例腳本，也可以根據需要更新它。

此指令碼顯示 Canary 部署，用於使用別名部署新狀態機版本。下列步驟概述指令碼執行的工作：

1. 如果publish_revision參數設置為 true，發布最新revision作為狀態機的下一個版本。如果部署成功，此版本將成為新的即時版本。

如果您設定publish_revision參數為 false 時，指令碼會部署狀態機器的最後發佈版本。

2. 如果別名尚不存在，請建立別名。如果別名不存在，請將此別名的 100% 流量指向新版本，然後結束指令碼。
3. 更新別名的路由配置，以將一小部分流量從舊版轉移到新版本。您可以設定此初期測試百分比canary_percentage參數。
4. 默認情況下，監視可配置CloudWatch每 60 秒發出一次警報。如果其中任何警示發出，請將 100% 的流量指向舊版，立即復原部署。

   在每個時間間隔之後 (以秒為單位)alarm_polling_interval，繼續監控警報。繼續監視，直到定義的時間間隔canary_interval_seconds已通過。
5. 如果期間未設定任何鬧鐘canary_interval_seconds，將 100% 的流量轉移到新版本。
6. 如果新版本部署成功，請刪除任何早於history_max參數。

```
#!/bin/bash
#
# AWS StepFunctions example showing how to create a canary deployment with a
# State Machine Alias and versions.
#
# Requirements: AWS CLI installed and credentials configured.
#
# A canary deployment deploys the new version alongside the old version, while
# routing only a small fraction of the overall traffic to the new version to
# see if there are any errors. Only once the new version has cleared a testing
# period will it start receiving 100% of traffic.
#
# For a Blue/Green or All at Once style deployment, you can set the
# canary_percentage to 100. The script will immediately shift 100% of traffic
# to the new version, but keep on monitoring the alarms (if any) during the
# canary_interval_seconds time interval. If any alarms raise during this period,
# the script will automatically rollback to the previous version.
#
# Step Functions allows you to keep a maximum of 1000 versions in version history
# for a state machine. This script has a version history deletion mechanism at
# the end, where it will delete any versions older than the limit specified.
#
# For a fuller example, that also demonstrates linear (or rolling) deployments,
# please see
```

```
# https://github.com/aws-samples/aws-stepfunctions-examples/blob/main/gradual-deploy/
sfndeploy.py

set -euo pipefail

# ************************************************************************
# you can safely change the variables in this block to your values
state_machine_name="my-state-machine"
alias_name="alias-1"
region="us-east-1"

# array of cloudwatch alarms to poll during the test period.
# to disable alarm checking, set alarm_names=()
alarm_names=("alarm1" "alarm name with a space")

# true to publish the current revision as the next version before deploy.
# false to deploy the latest version from the state machine's version history.
publish_revision=true

# true to force routing configuration update even if the current routing
# for the alias does not have a 100% routing config.
# false will abandon deploy attempt if current routing config not 100% to a
# single version.
# Be careful when you combine this flag with publish_revision - if you just
# rerun the script you might deploy the newly published revision from the
# previous run.
force=false

# percentage of traffic to route to the new version during the test period
canary_percentage=10

# how many seconds the canary deployment lasts before full deploy to 100%
canary_interval_seconds=300

# how often to poll the alarms
alarm_polling_interval=60

# how many versions to keep in history. delete versions prior to this.
# set to 0 to disable old version history deletion.
history_max=0
# ************************************************************************

######################################
# Update alias routing configuration.
```

```
#
# If you don't specify version 2 details, will only create 1 routing entry. In
# this case the routing entry weight must be 100.
#
# Globals:
#   alias_arn
# Arguments:
#   1. version 1 arn
#   2. version 1 weight
#   3. version 2 arn (optional)
#   4. version 2 weight (optional)
#######################################
function update_routing() {
  if [[ $# -eq 2 ]]; then
    local routing_config="[{\"stateMachineVersionArn\": \"$1\", \"weight\":$2}]"
  elif [[ $# -eq 4 ]]; then
    local routing_config="[{\"stateMachineVersionArn\": \"$1\", \"weight\":$2},
 {\"stateMachineVersionArn\": \"$3\", \"weight\":$4}]"
  else
    echo "You have to call update_routing with either 2 or 4 input arguments." >&2
    exit 1
  fi

  ${aws} update-state-machine-alias --state-machine-alias-arn ${alias_arn} --routing-
configuration "${routing_config}"
}

# ******************************************************************************
# pre-run validation
if [[ (("${#alarm_names[@]}" -gt 0)) ]]; then
  alarm_exists_count=$(aws cloudwatch describe-alarms --alarm-names "${alarm_names[@]}"
 --alarm-types "CompositeAlarm" "MetricAlarm" --query "length([MetricAlarms,
 CompositeAlarms][])" --output text)

  if [[ (("${#alarm_names[@]}" -ne "${alarm_exists_count}")) ]]; then
    echo All of the alarms to monitor do not exist in CloudWatch: $(IFS=,; echo
 "${alarm_names[*]}") >&2
    echo Only the following alarm names exist in CloudWatch:
    aws cloudwatch describe-alarms --alarm-names "${alarm_names[@]}" --alarm-types
 "CompositeAlarm" "MetricAlarm" --query "join(', ', [MetricAlarms, CompositeAlarms]
[].AlarmName)" --output text
    exit 1
  fi
fi
```

```
if [[ (("${history_max}" -gt 0)) && (("${history_max}" -lt 2)) ]]; then
  echo The minimum value for history_max is 2. This is the minimum number of older
 state machine versions to be able to rollback in the future. >&2
  exit 1
fi
# ********************************************************************************
# main block follows

account_id=$(aws sts get-caller-identity --query Account --output text)

sm_arn="arn:aws:states:${region}:${account_id}:stateMachine:${state_machine_name}"

# the aws command we'll be invoking a lot throughout.
aws="aws stepfunctions"

# promote the latest revision to the next version
if [[ "${publish_revision}" = true ]]; then
  new_version=$(${aws} publish-state-machine-version --state-machine-arn=$sm_arn --
query stateMachineVersionArn --output text)
  echo Published the current revision of state machine as the next version with arn:
 ${new_version}
else
  new_version=$(${aws} list-state-machine-versions --state-machine-arn ${sm_arn} --max-
results 1 --query "stateMachineVersions[0].stateMachineVersionArn" --output text)
  echo "Since publish_revision is false, using the latest version from the state
 machine's version history: ${new_version}"
fi

# find the alias if it exists
alias_arn_expected="${sm_arn}:${alias_name}"
alias_arn=$(${aws} list-state-machine-aliases --state-machine-arn
 ${sm_arn} --query "stateMachineAliases[?stateMachineAliasArn==\`
${alias_arn_expected}\`].stateMachineAliasArn" --output text)

if [[ "${alias_arn_expected}" == "${alias_arn}" ]]; then
  echo Found alias ${alias_arn}

  echo Current routing configuration is:
  ${aws} describe-state-machine-alias --state-machine-alias-arn "${alias_arn}" --query
 routingConfiguration
else
  echo Alias does not exist. Creating alias ${alias_arn_expected} and routing 100%
 traffic to new version ${new_version}
```

```
  ${aws} create-state-machine-alias --name "${alias_name}" --routing-configuration
 "[{\"stateMachineVersionArn\": \"${new_version}\", \"weight\":100}]"

  echo Done!
  exit 0
fi

# find the version to which the alias currently points (the current live version)
old_version=$(${aws} describe-state-machine-alias --state-machine-alias-arn $alias_arn
 --query "routingConfiguration[?weight==\`100\`].stateMachineVersionArn" --output text)

if [[ -z "${old_version}" ]]; then
  if [[ "${force}" = true ]]; then
    echo Force setting is true. Will force update to routing config for alias to point
 100% to new version.
    update_routing "${new_version}" 100

    echo Alias ${alias_arn} now pointing 100% to ${new_version}.
    echo Done!
    exit 0
  else
    echo Alias ${alias_arn} does not have a routing config entry with 100% of the
 traffic. This means there might be a deploy in progress, so not starting another
 deploy at this time. >&2
    exit 1
  fi
fi

if [[ "${old_version}" == "${new_version}" ]]; then
  echo The alias already points to this version. No update necessary.
  exit 0
fi

echo Switching ${canary_percentage}% to new version ${new_version}
(( old_weight = 100 - ${canary_percentage} ))
update_routing "${new_version}" ${canary_percentage} "${old_version}" ${old_weight}

echo New version receiving ${canary_percentage}% of traffic.
echo Old version ${old_version} is still receiving ${old_weight}%.

if [[ ${#alarm_names[@]} -eq 0 ]]; then
  echo No alarm_names set. Skipping cloudwatch monitoring.
```

```
  echo Will sleep for ${canary_interval_seconds} seconds before routing 100% to new
 version.
  sleep ${canary_interval_seconds}
  echo Canary period complete. Switching 100% of traffic to new version...
else
  echo Checking if alarms fire for the next ${canary_interval_seconds} seconds.

  (( total_wait = canary_interval_seconds + $(date +%s) ))

  now=$(date +%s)
  while [[ ((${now} -lt ${total_wait})) ]]; do
    alarm_result=$(aws cloudwatch describe-alarms --alarm-names "${alarm_names[@]}"
 --state-value ALARM --alarm-types "CompositeAlarm" "MetricAlarm" --query "join(', ',
 [MetricAlarms, CompositeAlarms][].AlarmName)" --output text)

    if [[ ! -z "${alarm_result}" ]]; then
      echo The following alarms are in ALARM state: ${alarm_result}. Rolling back
 deploy. >&2
      update_routing "${old_version}" 100

      echo Rolled back to ${old_version}
      exit 1
    fi

    echo Monitoring alarms...no alarms have triggered.
    sleep ${alarm_polling_interval}
    now=$(date +%s)
  done

  echo No alarms detected during canary period. Switching 100% of traffic to new
 version...
fi

update_routing "${new_version}" 100

echo Version ${new_version} is now receiving 100% of traffic.

if [[ (("${history_max}" -eq 0 ))]]; then
  echo Version History deletion is disabled. Remember to prune your history, the
 default limit is 1000 versions.
  echo Done!
  exit 0
fi
```

```
echo Keep the last ${history_max} versions. Deleting any versions older than that...

# the results are sorted in descending order of the version creation time
version_history=$(${aws} list-state-machine-versions --state-
machine-arn ${sm_arn} --max-results 1000 --query "join(\`\"\\n\"\`,
 stateMachineVersions[].stateMachineVersionArn)" --output text)

counter=0

while read line; do
  ((counter=${counter} + 1))

  if [[ (( ${counter} -gt ${history_max})) ]]; then
    echo Deleting old version ${line}
    ${aws} delete-state-machine-version --state-machine-version-arn ${line}
  fi
done <<< "${version_history}"

echo Done!
```

使用AWS用於部署新狀態機版本的 SDK

範例指令碼位於aws-stepfunctions-examples顯示如何使用AWS適用於 Python 的 SDK 可逐步將流量從舊版本轉移到新版本的狀態機器。您可以使用此示例腳本，也可以根據需要更新它。

此指令碼會顯示下列部署策略：

- 金絲雀— 以兩種增量轉移流量。

  在第一個增量中，一小部分的流量，例如，10% 會轉移到新版本。在第二個增量中，在以秒為單位的指定時間間隔結束之前，剩餘的流量會轉移到新版本。切換到新版本的剩餘流量發生，只有當沒有CloudWatch鬧鐘會在指定的時間間隔內設定。

- 線性或滾動— 以相等的增量將流量轉移到新版本，每個增量之間的秒數相等。

  例如，如果您將增量百分比指定為**20**用一個`--interval`的**600**秒，此部署每 600 秒會增加 20% 的流量，直到新版本收到 100% 的流量為止。

  此部署會立即復原新版本 (如果有的話)CloudWatch警報被設置。

- 一旦全部或藍色/綠色— 立即將 100% 的流量轉移到新版本。此部署會監控新版本，並自動將其復原至先前的版本 (如果有的話)CloudWatch警報被設置。

使用AWS CloudFormation部署新的狀態機版本

以下CloudFormation範本範例發佈名為的狀態機器的兩個版本*MyStateMachine*。它創建一個名為別名*PROD*，它指向這兩個版本，然後部署該版本2。

在此範例中，10% 的流量會轉移至版本2每五分鐘，直到此版本收到 100% 的流量。這個例子還顯示了如何設置CloudWatch警報。如果您設定的任何鬧鐘進入ALARM狀態，部署失敗並立即復原。

```
MyStateMachine:
  Type: AWS::StepFunctions::StateMachine
  Properties:
    Type: STANDARD
    StateMachineName: MyStateMachine
    RoleArn: arn:aws:iam::123456789012:role/myIamRole
    Definition:
      StartAt: PassState
      States:
        PassState:
          Type: Pass
          Result: Result
          End: true

MyStateMachineVersionA:
  Type: AWS::StepFunctions::StateMachineVersion
  Properties:
    Description: Version 1
    StateMachineArn: !Ref MyStateMachine

MyStateMachineVersionB:
  Type: AWS::StepFunctions::StateMachineVersion
  Properties:
    Description: Version 2
    StateMachineArn: !Ref MyStateMachine

PROD:
  Type: AWS::StepFunctions::StateMachineAlias
  Properties:
    Name: PROD
    Description: The PROD state machine alias taking production traffic.
    DeploymentPreference:
      StateMachineVersionArn: !Ref MyStateMachineVersionB
      Type: LINEAR
      Percentage: 10
```

```
      Interval: 5
      Alarms:
        # A list of alarms that you want to monitor. If any of these alarms trigger,
rollback the deployment immediately by pointing 100 percent of traffic to the previous
version.
        - !Ref CloudWatchAlarm1
        - !Ref CloudWatchAlarm2
```

Step Functions 數中的執行

當狀態機器執行並執行其工作時，就會發生AWS Step Functions狀態機器執行。每個步驟函數狀態機器可以有多個同時執行，您可以從 Step Functions 主控台或使用 AWS SDK、Step Functions API 動作或 AWS Command Line Interface () AWS CLI 來啟動這些執行。執行會收到 JSON 輸入並產生 JSON 輸出。您可以通過以下方式啟動 Step Functions 執行：

- 呼叫 StartExecution API 動作。
- 在「Step Functions」主控台中啟動新的執行。
- 使用 Amazon EventBridge 開始執行以回應事件。
- 用Amazon EventBridge Scheduler於按排程啟動狀態機器執行。
- 使用 Amazon API Gateway 開始執行。
- 從「工作」狀態啟動巢狀工作流程執行。

如需有關使用 Step Functions 之不同方式的詳細資訊，請參閱開發選項。

從任務狀態開始工作流程執行

AWS Step Functions 可以直接從狀態機器的 Task 狀態啟動工作流程執行。這可讓您將工作流程分成較小的狀態機器，並啟動這些其他狀態機器的執行。透過啟動這些新的工作流程執行，您可以：

- 將較高層級的工作流程與較低層級的特定工作流程分開。
- 多次呼叫不同的狀態機器，以避免重複元素。
- 建立模組化可重複使用工作流程的程式庫，以加快開發速度。
- 降低複雜性，並讓您更輕鬆地編輯狀態機器和排除其問題。

Step Functions 可以透過呼叫自己的 API 做為整合式服務來啟動這些工作流程執行。只需從您的 Task 狀態呼叫 StartExecution API 動作，並傳遞必要的參數即可。您可以使用任何服務整合模式呼叫步驟函式 API。

> **Tip**
>
> 若要將巢狀工作流程的範例部署到您的AWS 帳戶，請參閱單元 13-巢狀 Express 工作流程。

若要啟動狀態機器的新執行，請使用類似下列範例的Task狀態：

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution",
   "Parameters":{
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Input":{
         "Comment":"Hello world!"
      },
   },
   "Retry":[
      {
        "ErrorEquals":[
            "StepFunctions.ExecutionLimitExceeded"
        ]
      }
   ],
   "End":true
}
```

此 Task 狀態將啟動新的 HelloWorld 狀態機器執行，並傳遞 JSON 評論做為輸入。

> **Note**
>
> StartExecution API 動作配額會限制您可以啟動的執行數。在 StepFunctions.ExecutionLimitExceeded 上使用 Retry，以確保您的執行已啟動。請查看以下內容。
>
> - 與 API 動作節流相關的配額

- Step Functions 中的錯誤處理

關聯工作流程執行

若要將啟動的工作流程執行與啟動它的執行建立關聯，請將執行 ID 從內容物件傳遞至執行輸入。您可以從執行中執行的 Task 狀態中存取內容物件中的 ID。透過將 .$ 附加至參數名稱，並使用 $$.Execution.Id 參考內容物件中的 ID 來傳遞執行 ID。

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

您可以在啟動執行時使用名為 AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID 的特殊參數。如果包含此關聯，此關聯會在 Step Functions 主控台的「步驟詳細資訊」區段中提供連結。一旦提供，您便可輕鬆地追蹤工作流程的執行，從正在啟動的執行到已啟動的工作流程執行。使用先前的範例，將執行 ID 與 HelloWorld 狀態機器已啟動的執行建立關聯，如下所示。

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution",
   "Parameters":{
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Input": {
        "Comment": "Hello world!",
        "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
       }
   },
   "End":true
}
```

如需詳細資訊，請參閱下列內容：

- 使用其他 服務
- 將參數傳遞至服務 API
- 存取內容物件
- AWS Step Functions

使用亞馬遜 EventBridge 排程器 AWS Step Functions

Amazon EventBridge Scheduler 是無伺服器排程器，可讓您從單一中央受管服務建立、執行和管理任務。使用 EventBridge Scheduler，您可以使用循環模式的 cron 和速率運算式來建立排程，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。

例如，使用 EventBridge Scheduler，您可以在發生安全性相關事件時按排程啟動狀態機器執行，或將資料處理工作自動化。

本頁說明如何使用「 EventBridge 排程器」，按排程開始執行「Step Functions」狀態機器。

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呼叫其目標 API 作業的權限。您可以使用執行角色將這些權限授與 EventBridge 「排程器」。排程執行角色所連接的許可政策會定義哪些是必要許可。這些權限取決於您希望 EventBridge 排程器叫用的目標 API。

當您使用「 EventBridge 排程器」主控台建立排程時，如下列程序所示，「 EventBridge 排程器」會根據您選取的目標自動設定執行角色。如果您想要使用其中一個 S EventBridge cheduler SDK、或來建立排程AWS CloudFormation，您必須具有現有的執行角色，以授與 EventBridge 排程器呼叫目標所需的權限。AWS CLI如需有關手動設定排程執行角色的詳細資訊，請參閱《EventBridge 排程器使用指南》中的 < 設定執行角色 > 。

建立排程

使用主控台建立排程

1. 在 https://console.aws.amazon.com/scheduler/home 打開亞馬遜 EventBridge 調度程序控制台。
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 運算式定義排程，請選擇 Rate 排程，然後輸入 Rate 運算式。 如需 Cron 和費率運算式的詳細資訊，請參閱 Amazon 排程器使用者指南中的 EventBridge 排程器上的 EventBridge 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 排程器呼叫的 AWS API 作業：

   a. 選擇AWS Step Functions StartExecution。

   b. 在StartExecution區段中，選取狀態機或選擇建立新的狀態機。

      目前，您無法依排程執行「同步快速」工作流程。

   c. 輸入執行的 JSON 承載。即使您的狀態機器不需要任何 JSON 裝載，您仍然必須包含 JSON 格式的輸入，如下列範例所示。

```
{
    "Comment": "sampleJSONData"
}
```

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。
- 針對事件的保留時間上限，輸入 EventBridge 排程器必須保留未處理事件的最大小時數和最小時數。
- 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 排程器重試排程的次數上限。

  最大值為重試 185 次。

使用重試原則時，如果排程無法呼叫其目標， 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 排程器儲存未傳遞事件的位置。

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。 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 Scheduler 會將範本化目標所需的必要權限附加至角色。

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。

若要確認 EventBridge 排程器是否呼叫狀態機器，請檢查狀態機器的 Amazon CloudWatch 記錄。

相關資源

如需有關 EventBridge 排程器的詳細資訊，請參閱下列內容：

- EventBridge 排程器使用指南
- EventBridge 排程器 API 參考
- EventBridge 排程器定價

主控台中的標準和快速工作流程執行

建立狀態機時，選取「標準」或「快速」的「類型」。狀態機器的預設「類型」為「標準」。「類型」為「標準」的狀態機稱為「標準」工作流程，而「類型」為「快速」的狀態機稱為 Express 工作流程。

對於標準和 Express 工作流程，您可以使用定義狀態機器Amazon States Language。根據您選取的類型，狀態機器執行的行為會有所不同。

> **Important**
>
> 建立狀態機後，您選擇的「類型」無法變更。

如需有關「標準」和「快速」工作流程的更多資訊標準與快速工作流程，

標準工作流程執行的歷史記錄記錄在 Step Functions 中，而 Express 工作流程執行的歷史記錄不會記錄在 Step Functions 中。若要記錄 Express 工作流程執行的歷史記錄，您必須將其設定為將日誌傳送到 Amazon CloudWatch。如需詳細資訊，請參閱 記錄使用CloudWatch日誌。

在 Express 工作流程上設定記錄之後，您可以在 Step Functions 主控台中檢視其執行。檢視 Express 工作流程執行與標準工作流程執行的主控台體驗類似，但下列差異與限制除外。

> **Note**
>
> 由於 Express 工作流程的執行資料是使用 CloudWatch 記錄深入分析來顯示，因此掃描記錄會產生費用。根據預設，您的記錄群組只會列出過去三小時內完成的執行項目。如果您指定包含更多執行事件的較大時間範圍，成本就會增加。如需詳細資訊，請參閱定價頁面上 [記錄檔] 索引標籤下的 [CloudWatch 付費記錄] 和記錄使用CloudWatch日誌。

目錄

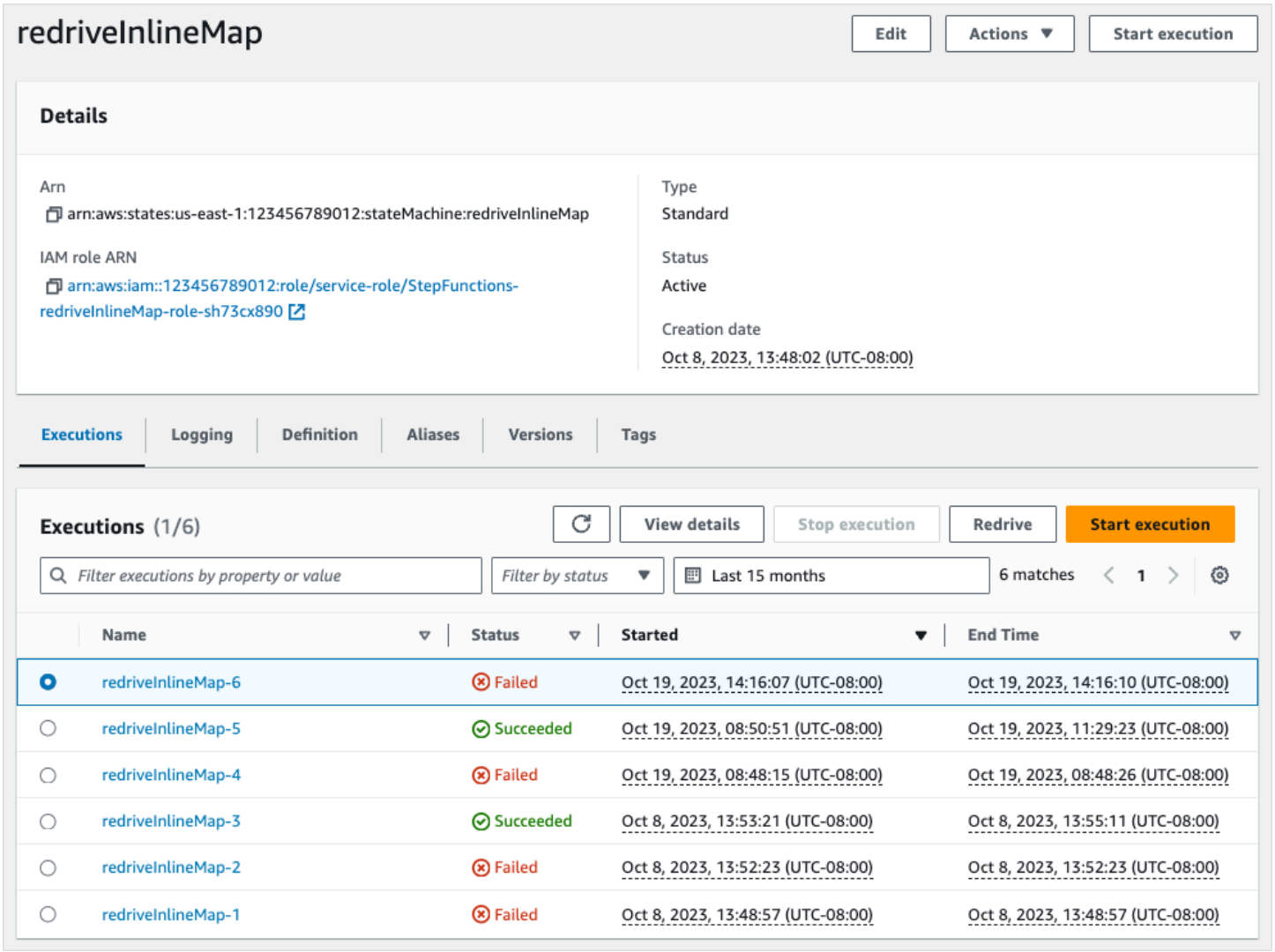
主控台體驗差異

對於所有標準和 Express 工作流程，您可以在 Step Functions 主控台的狀態機器詳細資料頁面上檢視詳細資料，例如狀態機器及其 IAM 角色 ARN。

在 [狀態機器詳細資料] 頁面上，您也可以在 [執行] 索引標籤下看到狀態機器的執行歷程記錄清單。使用 [依內容或值篩選執行項目] 方塊，搜尋所選狀態機器的特定執行項目、版本或別名。使用「全部」下拉式清單，依狀態篩選執行歷史記錄。您也可以選擇執行歷史記錄，然後選取檢視詳細資訊按鈕，開啟其「執行詳細資訊」頁面。

標準工作流

標準工作流程的執行歷史記錄始終可用於過去 90 天內完成的執行。



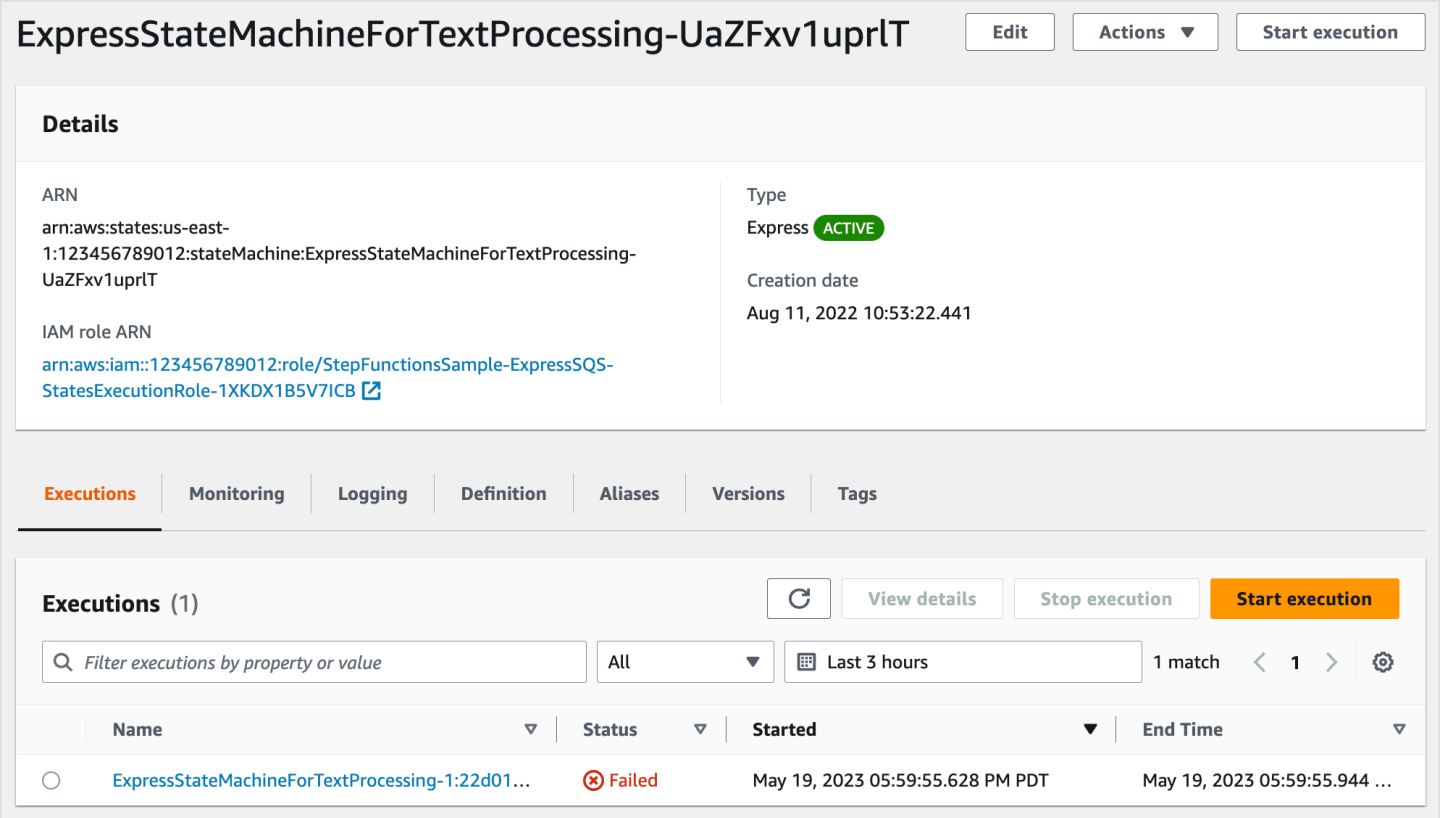
快速工作流

若要顯示 Express 工作流程的執行歷程記錄，Step Functions 主控台會擷取透過記錄檔記錄群組收集的 CloudWatch 記錄資料。

您也必須啟用新的主控台體驗，才能檢視 Express 工作流程執行。若要這麼做，請選擇 [執行] 索引標籤上標題內顯示的 [啟用] 按鈕。一旦您選擇此按鈕，它就不會再出現。

> **Tip**
>
> 若要在啟用或停用主控台體驗之間切換，請使用 [啟用快速執行歷程記錄] 切換按鈕。

依預設，可使用過去三小時內完成之執行項目的歷史記錄。您可以調整此時間範圍或指定自訂範圍。如果您指定包含更多執行事件的時間範圍較大，則掃描記錄檔的成本將會增加。如需詳細資訊，請參閱定價頁面上 [記錄檔] 索引標籤下的 [CloudWatch 付費記錄] 和記錄使用CloudWatch日誌。



檢視 Express 工作流程執行的考量與限制

在「Step Functions」主控台上檢視 Express 工作流程執行時，請記住下列考量和限制。

- 快速工作流程執行細節的可用性取決於 Amazon CloudWatch 日誌
- 如果記錄層級為「錯誤」或「嚴重」，則可使用部分快速工作流程
- 一旦更新，就無法查看較舊執行的狀態機定義

快速工作流程執行細節的可用性取決於 Amazon CloudWatch 日誌

> **Note**
>
> 如果您未啟用新的主控台體驗來檢視 Express 工作流程執行，則在 Step Functions 主控台中無法使用執行歷史記錄及其對應的執行詳細資料。若要啟用新的主控台體驗，請選擇 [執行] 索引標籤上橫幅內顯示的 [啟用] 按鈕。

對於 Express 工作流程，其執行歷史記錄和詳細的執行資訊是透過 CloudWatch 日誌深入解析收集。此資訊會保留在您建立狀態機器時指定的 CloudWatch 記錄檔記錄群組中。狀態機器的執行歷程記錄會顯示在 Step Functions 主控台的 [執行] 索引標籤下。有關狀態機器每次執行的詳細資訊，會顯示在所選執行項目的 [執行詳細資訊] 頁面上。

> **Warning**
>
> 如果您刪除 Express 工作流程的 CloudWatch 記錄檔，它不會列在 [執行] 索引標籤下。

建議您使用 ALL 的預設記錄層級來記錄所有執行事件類型。您可以在編輯現有狀態機器時，視需要更新記錄層級。如需詳細資訊，請參閱 記錄使用CloudWatch日誌 及 日誌層級。

如果記錄層級為「錯誤」或「嚴重」，則可使用部分快速工作流程

依預設，Express 工作流程執行的記錄層級設定為「全部」。如果您變更記錄層級，已完成執行的執行歷程記錄和執行詳細資料將不會受到影響。不過，所有新的執行都會根據更新的記錄層級發出記錄檔。如需詳細資訊，請參閱 記錄使用CloudWatch日誌 及 日誌層級。

例如，如果您將記錄層級從 ALL 變更為 ERROR 或 FATAL，則 [Step Functions 式] 主控台上的 [執行] 索引標籤只會列出失敗的執行項目。在 [事件檢視] 索引標籤中，主控台只會顯示失敗的狀態機器步驟的事件詳細資料。

建議您使用 ALL 的預設記錄層級來記錄所有執行事件類型。編輯狀態機時，您可以視需要更新現有狀態機器的記錄層級。

一旦更新，就無法查看較舊執行的狀態機定義

Express 工作流程不會儲存過去執行的狀態機器定義。如果您變更狀態機定義，則只能檢視使用最新定義之執行的狀態機器定義。

例如，如果您從狀態機器定義中移除一或多個步驟，Step Functions 會偵測定義與先前執行事件之間的不相符。由於先前的定義不會儲存 Express 工作流程，因此 Step Functions 無法顯示在舊版狀態機器定義上執行的狀態機器定義。因此，在舊版狀態機器定義上執行的執行無法使用 [執行輸入和輸出]、[定義]、[圖形檢視] 和 [表格檢視] 索引標籤。

在 Step Functions 主控台上檢視和偵錯執行

「Step Functions」主控台上的「執行詳細資訊」頁面會顯示標準和 Express 工作流程過去和進行中狀態機器執行的相關資訊。此資訊會以儀表板格式顯示。例如，您可以找到狀態機器的 Amazon 狀態語

言定義、其執行狀態、ARN 以及狀態轉換的總數。您也可以檢視狀態機器中任何個別狀態的執行詳細資料。

目錄

執行詳細資訊頁面 — 介面概觀

您可以在 [執行詳細資訊] 頁面上找到標準工作流程和 Express 工作流程的所有進行中和過去狀態機器執行的詳細資訊。如果您在開始執行時指定了執行 ID，則此頁面的標題為該執行 ID。否則，它的標題為 Step Functions 自動為您生成的唯一執行 ID。

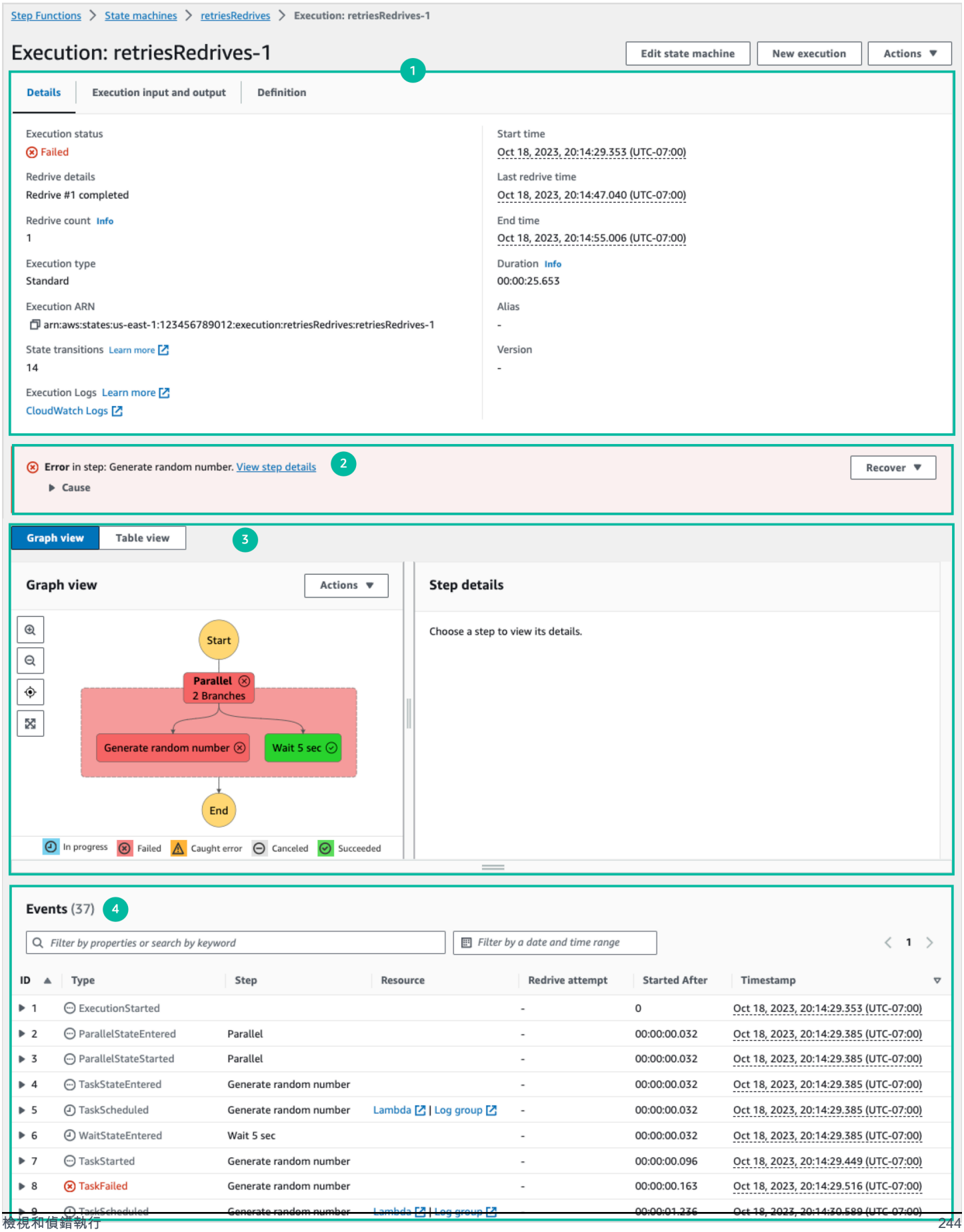
除了執行測量結果之外，「執行詳細資訊」頁面還提供下列選項，用於管理狀態機器及其執行：

按鈕	選擇此按鈕可：
編輯狀態機	編輯狀態機器的 Amazon 州語言定義。
新的執行	啟動狀態機的新執行。
動作	提供下列選項供您選擇： • 停止執行 — 停止進行中的執行。對於已完成的執行，此選項無法使用。 • Redrive— 在過去 14 天內未成功完成的標準工作流程的Redrive執行。其中包括失敗、

按鈕	選擇此按鈕可：
	中止或逾時的執行。如需詳細資訊，請參閱 Redriving處決。 • 匯出 — 以 JSON 格式匯出執行詳細資訊，以便與他人共用或執行離線分析。 • 傳送意見反應 — 分享有關介面的意見反應。

> **檢視以版本或別名開始的執行**
>
> 您也可以在 Step Functions 主控台中檢視以版本或別名開始的執行。如需詳細資訊，請參閱列出版本和別名的執行項目。

「執行詳細資訊：主控台」頁面包含下列段落：


Step Functions > State machines > retriesRedrives > Execution: retriesRedrives-1
Execution: retriesRedrives-1
Edit state machine
New execution
Actions
1
Details
Execution input and output
Definition
Execution status
Failed
Redrive details
Redrive #1 completed
Redrive count Info
1
Execution type
Standard
Execution ARN
arn:aws:states:us-east-1:123456789012:execution:retriesRedrives:retriesRedrives-1
State transitions Learn more
14
Execution Logs Learn more
CloudWatch Logs
Start time
Oct 18, 2023, 20:14:29.353 (UTC-07:00)
Last redrive time
Oct 18, 2023, 20:14:47.040 (UTC-07:00)
End time
Oct 18, 2023, 20:14:55.006 (UTC-07:00)
Duration Info
00:00:25.653
Alias
-
Version
-
Error in step: Generate random number. View step details
2
Cause
Recover
Graph view
Table view
3
Graph view
Actions
Start
Parallel
2 Branches
Generate random number
Wait 5 sec
End
In progress
Failed
Caught error
Canceled
Succeeded
Step details
Choose a step to view its details.
Events (37)
4
Filter by properties or search by keyword
Filter by a date and time range
1
ID
Type
Step
Resource
Redrive attempt
Started After
Timestamp
1 ExecutionStarted - 0 Oct 18, 2023, 20:14:29.353 (UTC-07:00)
2 ParallelStateEntered Parallel - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
3 ParallelStateStarted Parallel - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
4 TaskStateEntered Generate random number - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
5 TaskScheduled Generate random number Lambda | Log group - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
6 WaitStateEntered Wait 5 sec - 00:00:00.032 Oct 18, 2023, 20:14:29.385 (UTC-07:00)
7 TaskStarted Generate random number - 00:00:00.096 Oct 18, 2023, 20:14:29.449 (UTC-07:00)
8 TaskFailed Generate random number - 00:00:00.163 Oct 18, 2023, 20:14:29.516 (UTC-07:00)
9 TaskScheduled Generate random number Lambda | Log group - 00:00:01.236 Oct 18, 2023, 20:14:30.589 (UTC-07:00)

1. 執行摘要
2. 錯誤訊息
3. 檢視模式
4. 步驟詳情
5. 事件

執行摘要

「執行項目摘要」段落會顯示在「執行詳細資訊」頁面的頂端。本節提供工作流程執行詳細資訊的概觀。此資訊分為以下三個標籤：

詳細資訊

顯示執行開始和結束時間的資訊，例如執行狀態、ARN 和時間戳記。您也可以檢視執行狀態機器執行時所發生的「狀態」轉換總數。如果您已為狀態機器啟用追蹤或記錄，也可以檢視 X-Ray 追蹤對應和 Amazon CloudWatch 執行日誌的連結。

如果您的狀態機執行是由另一個狀態機器啟動的，您可以在此索引標籤上檢視父狀態機器的連結。

如果您的狀態機器執行是 redriven，此索引標籤會顯示redrive相關資訊，例如 Redrivecount。

執行輸入和輸出

顯示狀態機執行輸入和輸出 side-by-side。

定義

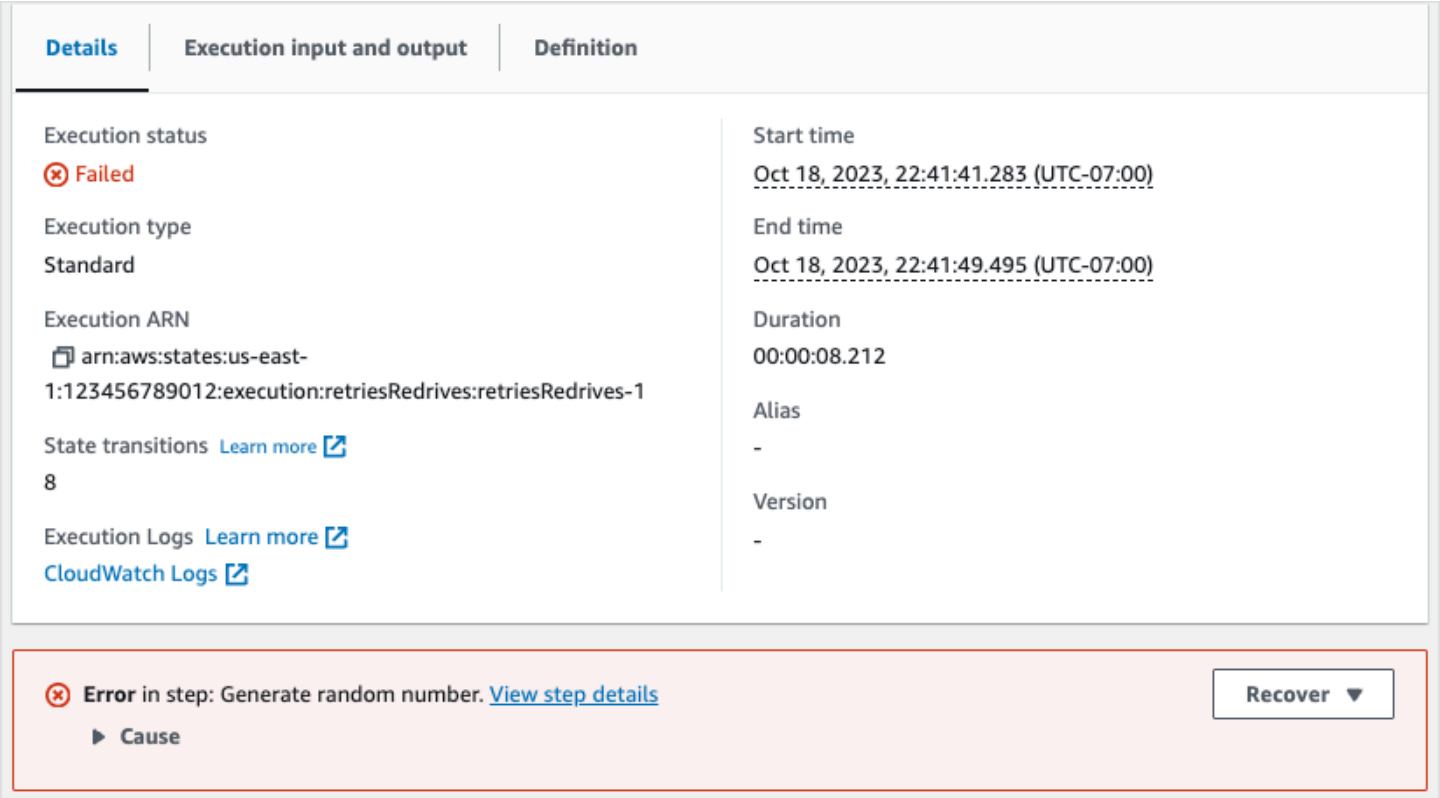
顯示狀態機器的 Amazon 州語言定義。

錯誤訊息

如果狀態機器執行失敗，[執行詳細資訊] 頁面會顯示錯誤訊息。在錯誤訊息中選擇「原因」或「檢視步驟詳細資訊」，以檢視執行失敗的原因或造成錯誤的步驟。

如果您選擇檢視步驟詳細資訊，「Step Functions」會在「步驟詳細資訊」、「圖形檢視」及「表格檢視」標籤中反白顯示造成錯誤的步驟。如果步驟是您已為其定義重試的 [工作]、[對映] 或 [平行] 狀態，[步驟] 詳細資料窗格會顯示該步驟的 [重試] 索引標籤。此外，如果您已redriven執行，您可以在 [步驟] 詳細資料窗格的 [重試與redrives] 索引標籤中看到重試和redrive執行詳細資料。

從此錯誤消息上的「恢復」下拉按鈕中，redrive您可以執行失敗或開始新的執行。如需詳細資訊，請參閱 Redriving處決。



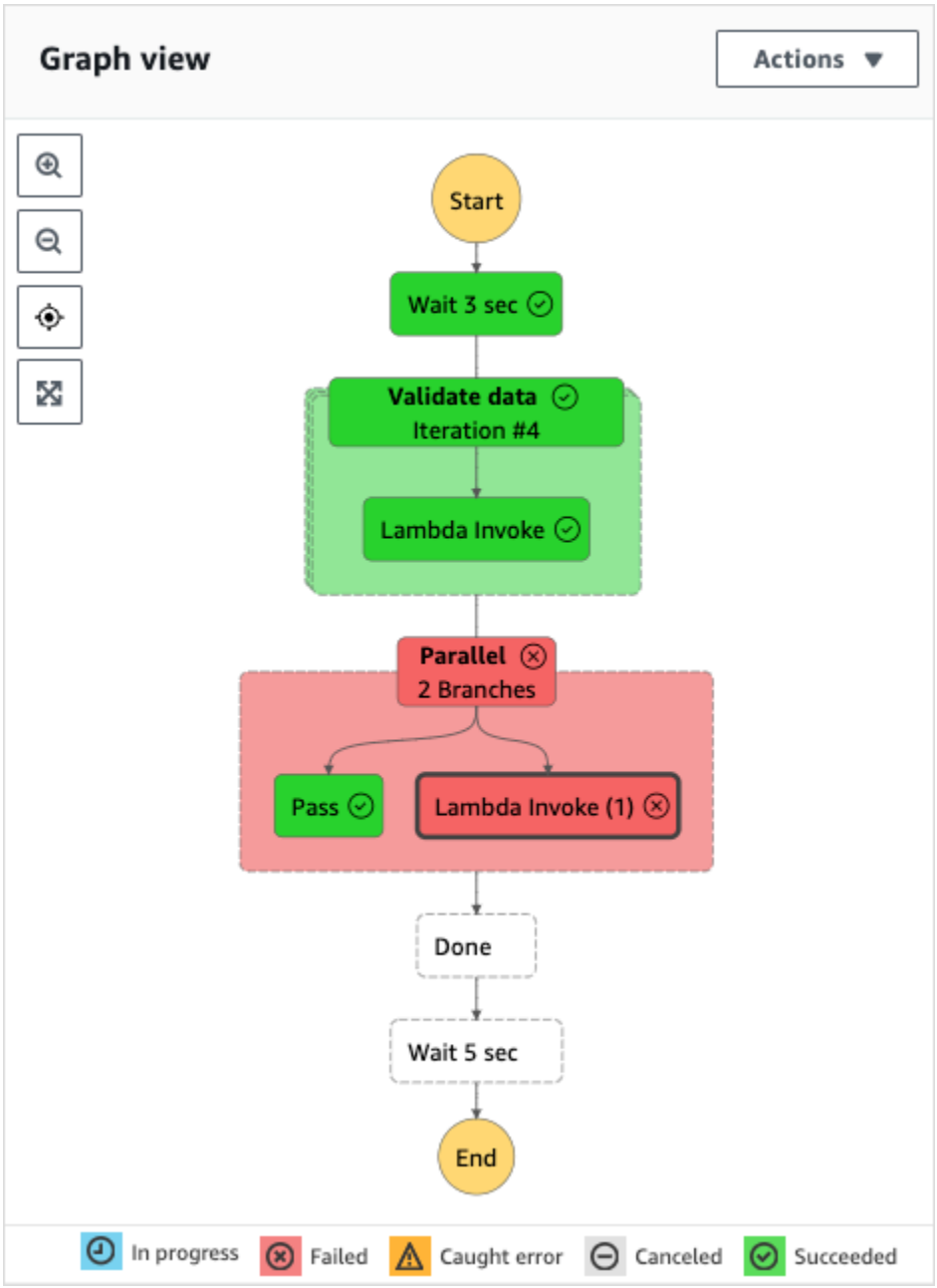
檢視模式

[檢視模式] 區段包含狀態機器的兩種不同視覺效果。您可以選擇檢視工作流程的圖形表示、概述工作流程中狀態的表格，或是與狀態機器執行相關聯的事件清單：

> **Note**
>
> 選擇索引標籤以檢視其內容。

Graph view

「圖形」檢視模式會顯示工作流程的圖形表示。底部包含一個圖例，指示狀態機的執行狀態。它也包含可讓您放大、縮小、置中對齊完整工作流程或以全螢幕模式檢視工作流程的按鈕。

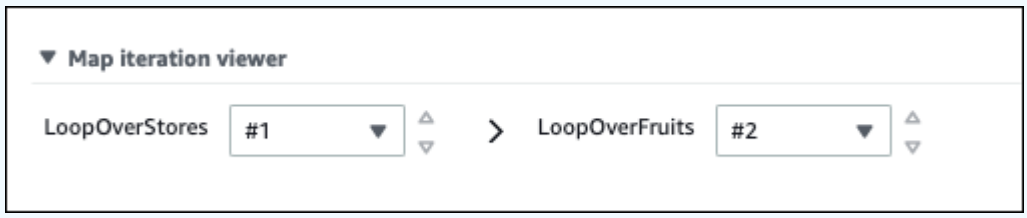


從此檢視中，您可以選擇工作流程中的任何步驟，在「步驟詳細資料」元件中檢視其執行的詳細資訊。當您在「圖形」檢視中選擇步驟時，「表格」檢視也會顯示該步驟。這也是相反的。如果您從「表格」檢視中選擇步驟，「圖表」檢視會顯示相同的步驟。

如果您的狀態機包含Map狀態、Parallel狀態或兩者，您可以在「圖形」檢視的工作流程中檢視它們的名稱。此外，對於Map狀態，「圖形」檢視可讓您在 Map 狀態執行資料的不同版序之間移動。例如，如果您的 Map 狀態有五個反覆項目，而您想要檢視第三次和第四個反覆項目的執行資料，請執行下列動作：

1. 選擇您要檢視其版序資料的「對映」狀態。
2. 從「地圖版序檢視器」中，從下拉式清單中選擇 #2 以進行第三個版序。這是因為迭代從零開始計數。同樣地，從下拉式清單中選擇 #3 做為 Map 狀態的第四次迭代。

   或者，使
   用△ 和▽
   制項在 Map 狀態的不同版序之間移動。

   > **Note**
   >
   > 如果您的狀態機器包含巢Map狀狀態，則會顯示父和子Map狀態版序的下拉式清單，如下列範例所示：
   >
   > 
   >

3. （可選）如果一個或多個 Map 狀態迭代無法執行，或者執行已停止，則可以通過在下拉列表中的「失敗」或「中止」下選擇迭代號來查看其數據。

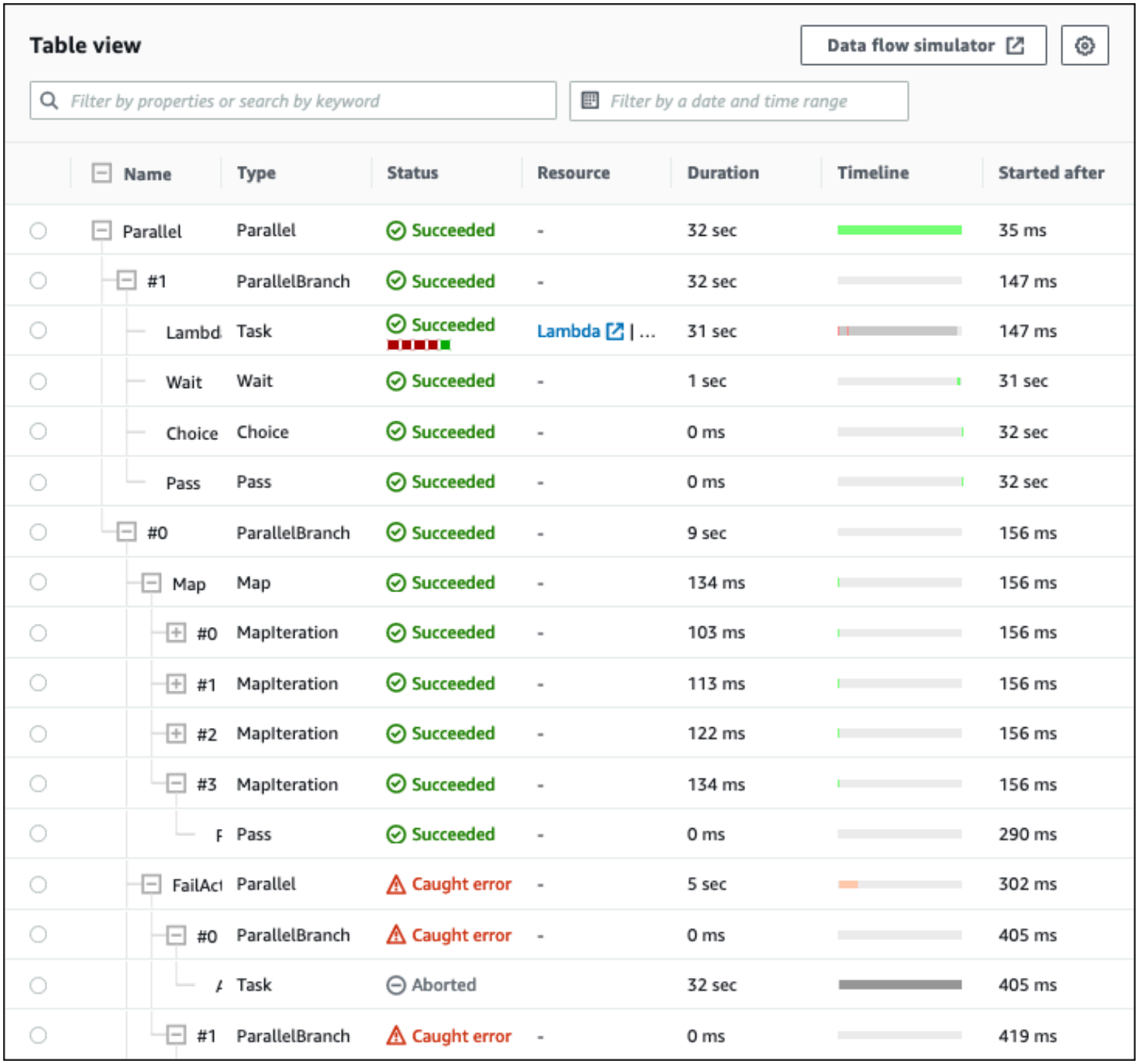
最後，您可以使用 [匯出] 和 [版面配置] 按鈕將工作流程圖形匯出為 SVG 或 PNG 影像。您也可以在工作流程的水平和垂直檢視之間切換。

Table view

「表格」檢視模式會顯示工作流程中狀態的表格表示法。在此「檢視」模式中，您可以查看在工作流程中執行的每個狀態的詳細資訊，包括其名稱、使用的任何資源名稱 (例如 AWS Lambda 函數)，以及狀態是否成功執行。

從此檢視中，您可以選擇工作流程中的任何狀態，以在「步驟詳細資料」元件中檢視其執行的詳細資訊。當您在「表格」檢視中選擇步驟時，「圖表」檢視也會顯示該步驟。這也是相反的。如果您從「圖形」檢視中選擇步驟，「表格」檢視會顯示相同的步驟。

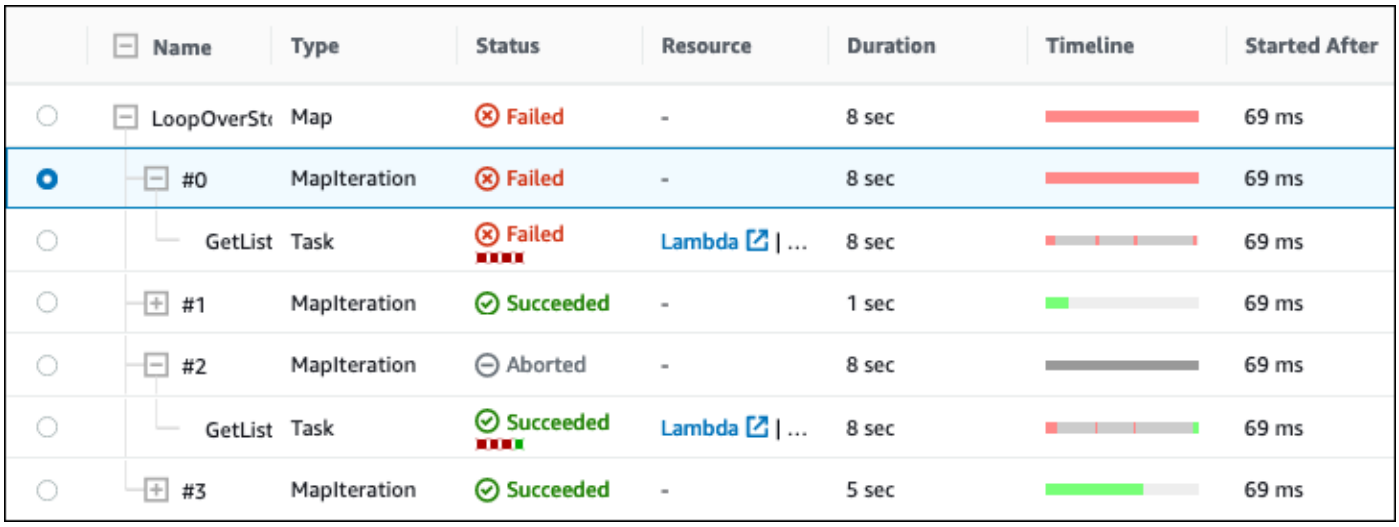
您也可以透過對檢視套用篩選條件來限制在「表格」檢視模式中顯示的資料量。您可以為特定性質 (例如 「狀態」或「Redrive嘗試」) 建立篩選器。如需詳細資訊，請參閱 教學課程：使用 Step Functions 主控台檢查狀態機器執行。



依預設，此模式會顯示 [名稱]、[類型]、[狀態]、[資源] 和 [在下列時間後開始] 欄 您可以使用「偏好設定」對話方塊來設定要檢視的欄。您在此對話方塊上所做的選擇會保留以 future 的狀態機器執行，直到它們再次變更為止。

如果您新增「時間軸」欄，每個狀態的執行持續時間會顯示與整個執行階段相關的執行持續時間。這會顯示為顏色編碼的線性時間線。這可以幫助您識別與特定狀態執行的任何效能相關問題。時間表上每個狀態的顏色區段可協助您識別狀態的執行狀態，例如進行中、失敗或中止。

例如，如果您已定義狀態機器中某個狀態的執行重試，則這些重試會顯示在計時列中。紅色區段代表失敗的Retry嘗試次數，而淺灰色區段代表每次Retry嘗試BackoffRate之間的次數。



如果您的狀態機包含Map狀態、狀Parallel態或兩者，您可以在 [表格] 檢視的工作流程中檢視它們的名稱。針對Map和Parallel狀態，「表格」檢視模式會將其迭代和 parallel 分支的執行資料顯示為樹狀檢視內的節點。您可以選擇處於這些狀態的每個節點，在「步驟詳細資訊」區段中檢視其個別詳細資訊。例如，您可以檢閱導致狀態失敗的特定 Map 狀態版序的資料。展開「對映」狀態的節點，然後在「狀況」欄中檢視每個版序的狀況。

步驟詳情

當您在「圖形」檢視或「表格」檢視中選擇狀態時，「步驟詳細資訊」區段會在右側開啟。此段落包含下列索引標籤，提供有關所選狀態的深入資訊：

輸入

顯示所選狀態的輸入詳細資訊。如果輸入中存在錯誤，則在標籤標題⚠以 a 表示。此外，您可以在此選項卡中查看錯誤的原因。

您也可以選擇 [進階檢視] 切換按鈕，以在資料通過所選狀態時查看輸入資料傳輸路徑。這可讓您識別如何將輸入作為一個或多個欄位 (例如InputPathParametersResultSelector、OutputPath、ResultPath、和) 套用至資料。

上

輸出

顯示所選狀態的輸出。如果輸出中有錯誤，則在標籤標題⚠上以 a 表示。此外，您可以在此選項卡中查看錯誤的原因。

您也可以選擇 [進階檢視] 切換按鈕，以在資料通過所選狀態時查看輸出資料傳輸路徑。這可讓您識別如何將輸入作為一個或多個欄位 (例如`InputPathParametersResultSelector`、`OutputPath`、`ResultPath`、和) 套用至資料。

詳細資訊

顯示資訊，例如狀態類型、其執行狀態和執行持續時間。

對於使用資源 (例如) 的`Task`狀態 AWS Lambda，此索引標籤會提供資源叫用的資源定義頁面和 Amazon CloudWatch 日誌頁面的連結。它也會顯示`Task`州/省`TimeoutSeconds`和`HeartbeatSeconds`欄位的值 (如果指定)。

對於`Map`狀態，此標籤會顯示`Map`狀態迭代總計數的相關資訊。版序會分類為「失敗」、「已中止」、「成功」或InProgress。

定義

顯示與所選州對應的 Amazon 州/省語言定義。

重試

> **Note**
>
> 只有當您在狀態機器`Task`或`Parallel`狀態中定義了`Retry`欄位時，才會顯示此索引標籤。

顯示所選狀態在其原始執行嘗試中的初始和後續重試嘗試。對於初始嘗試和所有後續的失敗嘗試，請選擇「類型」▶旁邊的，以檢視出現在下拉式方塊中的失敗原因。如果重試嘗試成功，您可以檢視出現在下拉式方塊中的輸出。

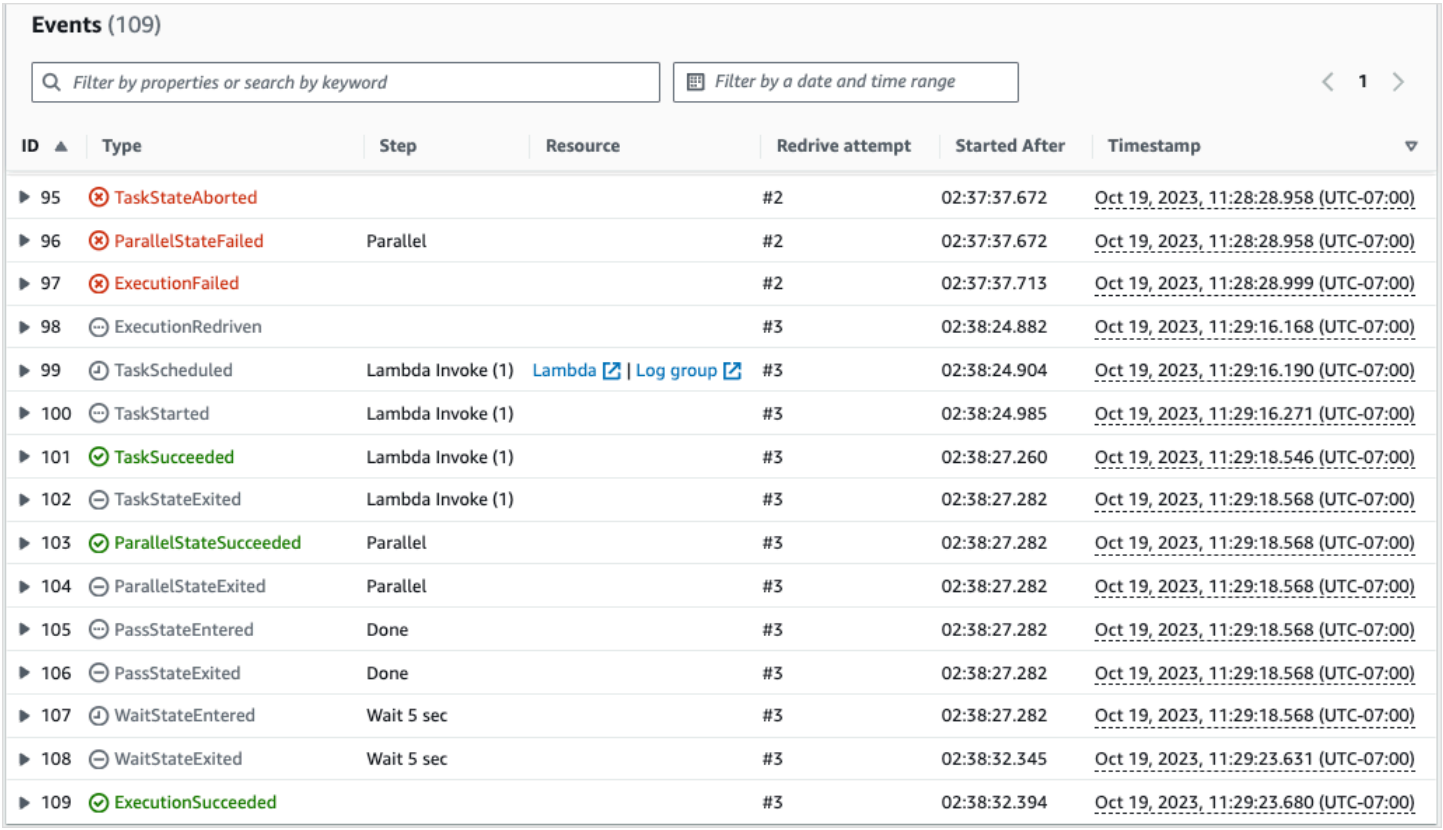
如果redriven您已執行，則此索引標籤標頭會顯示名稱「重試」(Retry &)，redrives並顯示每個redrive項目的重試嘗試詳細資料。

事件

顯示與執行中所選狀態相關聯之事件的篩選清單。您在此標籤上看到的資訊是您在「事件」(E vents) 表格中看到的完整執行事件歷程記錄的子集。

事件

「事件」表格會將所選執行項目的完整歷史記錄顯示為橫跨多個頁面的事件清單。每個頁面最多包含 25 個事件。此區段也會顯示事件總計數，以協助您判斷是否超過 25,000 個事件的最大事件歷程記錄計數。



依預設，「事件」(E vents) 表格中的結果會根據事件的時間戳記，以遞增順序顯示。您可以按一下「時間戳記」欄標題，將執行事件歷史記錄的排序變更為遞減順序。

在「事件」(Events) 表格中，每個事件都有顏色編碼，以指出其執行狀態 例如，失敗的事件會以紅色顯示。若要檢視有關事件的其他詳細資料，請選擇事件 ID ▶ 旁邊的。一旦開啟，事件詳細資訊就會顯示事件的輸入、輸出和資源叫用。

此外，在「事件」(E vents) 表格中，您可以套用篩選器來限制顯示的執行事件歷程記錄結果。您可以選擇屬性，例如 ID，或Redrive嘗試。如需更多詳細資訊，請參閱 教學課程：使用 Step Functions 主控台檢查狀態機器執行。

教學課程：使用 Step Functions 主控台檢查狀態機器執行

在本教學課程中，您將學習如何檢查 [執行詳細資訊] 頁面上顯示的執行資訊，並檢視執行失敗的原因。然後，您將學習如何訪問Map狀態執行的不同迭代。最後，您將學習如何在 [表格] 檢視上設定資料行，並套用適當的篩選器，以便只檢視您感興趣的資訊。

在本教程中，您將創建一個標準類型狀態機，該狀態機獲得一組水果的價格。為此，狀態機使用三個 AWS Lambda 函數，它們返回四個水果的隨機列表，每種水果的價格以及水果的平均成本。如果水果的價格小於或等於閾值，則 Lambda 函數旨在拋出錯誤。

> **Note**
>
> 下列程序包含如何檢查「標準」工作流程執行詳細資訊的指示，但您也可以檢查 Express 工作流程執行的詳細資訊。如需有關「標準」和「快速」工作流程類型執行詳細資訊之間差異的資訊，請參閱主控台中的標準和快速工作流程執行。

目錄

步驟 1：建立並測試所需的 Lambda 函數

1. 開啟 L ambda 主控台，然後執行步驟 1：建立 Lambda 函數本節中的步驟 1 到 4。確保命名 Lambda 函數**`GetListOfFruits`**。
2. 建立 Lambda 函數之後，請複製頁面右上角顯示的函數的 Amazon 資源名稱 (ARN)。若要複製 ARN，請按一下。 以
   下是 ARN 範例，其中*`function-name`*是 Lambda 函數的名稱 (在本例中為`GetListOfFruits`)：

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

3. 將 Lambda 函數的下列程式碼複製到GetListOfFruits頁面的程式碼原始碼區域。

```
function getRandomSubarray(arr, size) {
    var shuffled = arr.slice(0), i = arr.length, temp, index;
    while (i--) {
        index = Math.floor((i + 1) * Math.random());
        temp = shuffled[index];
        shuffled[index] = shuffled[i];
        shuffled[i] = temp;
    }
    return shuffled.slice(0, size);
}

exports.handler = async function(event, context) {

    const fruits = ['Abiu','Açaí','Acerola','Ackee','African
 cucumber','Apple','Apricot','Avocado','Banana','Bilberry','Blackberry','Blackcurrant','Jos


     const errorChance = 45;

    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
    // const num = 51;
     if (num <= errorChance) {
         throw(new Error('Error'));
     }

    return getRandomSubarray(fruits, 4);
};
```

4. 選擇 [部署]，然後選擇 [測試] 以部署變更並查看 Lambda 函數的輸出。
5. 使用下列步驟建立兩個**CalculateAverage**分別命名**GetFruitPrice**和的其他 Lambda 函數：

   a. 將下列程式碼複製到 GetFruitPriceLambda 函數的程式碼原始碼區域：

```
exports.handler = async function(event, context) {
```

```
    const errorChance = 0;
    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
    if (num <= errorChance) {
        throw(new Error('Error'));
    }

    return Math.floor(Math.random()*100)/10;
};
```

b. 將下列程式碼複製到 CalculateAverageLambda 函數的程式碼原始碼區域：

```
function getRandomSubarray(arr, size) {
    var shuffled = arr.slice(0), i = arr.length, temp, index;
    while (i--) {
        index = Math.floor((i + 1) * Math.random());
        temp = shuffled[index];
        shuffled[index] = shuffled[i];
        shuffled[i] = temp;
    }
    return shuffled.slice(0, size);
}

const average = arr => arr.reduce( ( p, c ) => p + c, 0 ) / arr.length;

exports.handler = async function(event, context) {
        const errors = [
        "Error getting data from DynamoDB",
        "Error connecting to DynamoDB",
        "Network error",
        "MemoryError - Low memory"
        ]

    const errorChance = 0;

    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
```

```
    if (num <= errorChance) {
        throw(new Error(getRandomSubarray(errors, 1)[0]));
    }

    return average(event);
};
```

c. 請務必複製這兩個 Lambda 函數的 ARN，然後進行部署和測試。

步驟 2：創建並執行狀態機

使用 Step Functions 主控台建立狀態機器，以叫用您在步驟 1 中建立的 Lambda 函數。在此狀態機中，定義了三種Map狀態。這些Map狀Task態中的每一個都包含會叫用其中一個 Lambda 函數的狀態。此外，在每個Task狀態中定義一個Retry欄位，其中包含針對每個狀態定義的重試次數。如果Task狀態遇到執行階段錯誤，它會再次執行，直到為此定義的重試嘗試次數為止Task。

1. 開啟 Step Functions 主控台，然後選擇 [以程式碼撰寫工作流程]。

> **Important**
>
> 確保您的狀態機與先前建立的 Lambda 函數位於相同的 AWS 帳戶和區域。

2. 對於「類型」，保留「標準」的預設選項。
3. 複製下列 Amazon 各州語言定義，並將其貼到「定義」下。請務必以先前建立的 Lambda 函數取代顯示的 ARN。

```
{
    "StartAt": "LoopOverStores",
    "States": {
        "LoopOverStores": {
            "Type": "Map",
            "Iterator": {
                "StartAt": "GetListOfFruits",
                "States": {
                    "GetListOfFruits": {
                        "Type": "Task",
                        "Resource": "arn:aws:states:::lambda:invoke",
                        "OutputPath": "$.Payload",
                        "Parameters": {
                            "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:GetListofFruits:$LATEST",
```

```
                            "Payload": {
                                "storeName.$": "$"
                            }
                        },
                        "Retry": [
                            {
                                "ErrorEquals": [
                                    "States.ALL"
                                ],
                                "IntervalSeconds": 2,
                                "MaxAttempts": 1,
                                "BackoffRate": 1.3
                            }
                        ],
                        "Next": "LoopOverFruits"
                    },
                    "LoopOverFruits": {
                        "Type": "Map",
                        "Iterator": {
                            "StartAt": "GetFruitPrice",
                            "States": {
                                "GetFruitPrice": {
                                    "Type": "Task",
                                    "Resource": "arn:aws:states:::lambda:invoke",
                                    "OutputPath": "$.Payload",
                                    "Parameters": {
                                        "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:GetFruitPrice:$LATEST",
                                        "Payload": {
                                            "fruitName.$": "$"
                                        }
                                    },
                                    "Retry": [
                                        {
                                            "ErrorEquals": [
                                                "States.ALL"
                                            ],
                                            "IntervalSeconds": 2,
                                            "MaxAttempts": 3,
                                            "BackoffRate": 1.3
                                        }
                                    ],
                                    "End": true
                                }
```

```
                            }
                        },
                        "ItemsPath": "$",
                        "End": true
                    }
                }
            },
            "ItemsPath": "$.stores",
            "Next": "LoopOverStoreFruitsPrice",
            "ResultPath": "$.storesFruitsPrice"
        },
        "LoopOverStoreFruitsPrice": {
            "Type": "Map",
            "End": true,
            "Iterator": {
                "StartAt": "CalculateAverage",
                "States": {
                    "CalculateAverage": {
                        "Type": "Task",
                        "Resource": "arn:aws:states:::lambda:invoke",
                        "OutputPath": "$.Payload",
                        "Parameters": {
                            "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:Calculate-average:$LATEST",
                            "Payload.$": "$"
                        },
                        "Retry": [
                            {
                                "ErrorEquals": [
                                    "States.ALL"
                                ],
                                "IntervalSeconds": 2,
                                "MaxAttempts": 2,
                                "BackoffRate": 1.3
                            }
                        ],
                        "End": true
                    }
                }
            },
            "ItemsPath": "$.storesFruitsPrice",
            "ResultPath": "$.storesPriceAverage",
            "MaxConcurrency": 1
        }
```

```
    }
}
```

4. 輸入狀態機的名稱。保留此頁面上其他選項的預設選項，然後選擇建立狀態機器。
5. 打開標題為您的狀態機名稱的頁面。執行步驟 4：運行狀態機區段中的步驟 1 到 4，但使用下列資料做為執行輸入：

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
      "Store D"
    ]
}
```

步驟 3：檢視狀態機執行詳細資料

在標題為執行 ID 的頁面上，您可以查看執行結果並對任何錯誤進行除錯。

1. (選擇性) 從「執行詳細資訊」頁面上顯示的標籤中選擇，以查看每個標籤中的資訊。例如，若要檢視狀態機器輸入及其執行輸出，請在 [執行摘要] 區段中選擇 [執行輸入和輸出]。
2. 如果狀態機執行失敗，請在錯誤訊息上選擇「原因」或「顯示步驟詳細資料」。有關錯誤的詳細資訊會顯示在「步驟詳細資訊」區段中。請注意，造成錯誤的步驟 (名為TaskGetListofFruits狀態) 會在「圖形」檢視和「表格」檢視中反白顯示。

> **Note**
>
> 由於GetListofFruits步驟是在狀Map態內定義的，且步驟無法順利執行，因此狀態步驟的Map「狀態」 步驟會顯示為「失敗」。

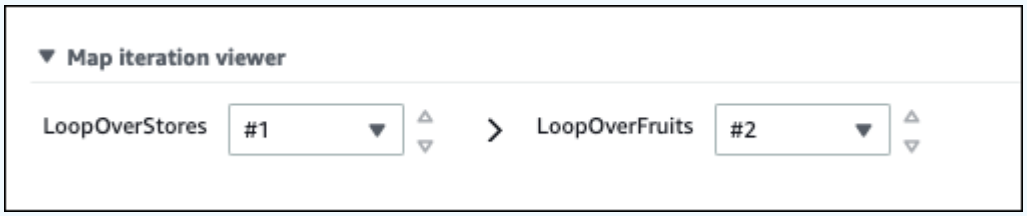
步驟 4：探索不同的「檢視」模式

您可以選擇偏好的模式來檢視狀態機工作流程或執行事件歷程記錄。您可以在這些「檢視」模式下執行的一些工作如下：

圖形檢視 — 在不同的**Map**狀態反覆之間切換

如果您的 Map 狀態有五個反覆項目，而您想要檢視第三個和第四個反覆項目的執行詳細資訊，請執行下列動作：

1. 選擇您要檢視其版序資料的Map狀態。
2. 從「對映版序檢視器」中，選擇您要檢視的版序。迭代從零開始計數。若要從五個版序中選擇第三個版序，請從「地圖」狀態名稱旁邊的下拉式清單中選擇 #2。

> **Note**
>
> 如果您的狀態機器包含嵌套Map狀態，則 Step Functions 會將父和子Map狀態迭代顯示為兩個單獨的下拉列表：
>
> 
>

3. (選擇性) 如果一或多個Map狀態版序無法執行或處於中止狀態停止，您可以檢視有關失敗版序的詳細資訊。若要查看這些詳細資訊，請在下拉式清單中的 [失敗] 或 [中止] 下選擇受影響的版序編號。

表格檢視 — 在不同**Map**狀態版序之間切換

如果您的 Map 狀態有五個版序，而您想要檢視版序號 3 和 4 的執行詳細資訊，請執行下列操作：

1. 選擇您要檢視其他版序資料的Map狀態。
2. 在狀態版序的樹Map狀檢視顯示中，針對第三版序選擇名為 #2 的版序列。同樣地，為版序號四選擇名為 #3 的列。

表格檢視 — 設定要顯示的欄

選擇

。

然後，在「偏好設定」對話方塊中，選擇要在「選取可見欄」下顯示的欄。

依預設，此模式會顯示 [名稱]、[類型]、[狀態]、[資源] 和 [在下列時間後開始] 欄

表格檢視 — 篩選結果

根據屬性 (例如「狀態」 或日期和時間範圍) 套用一或多個篩選條件，以限制顯示的資訊量。例如，若要檢視執行失敗的步驟，請套用下列篩選器：

1. 選擇「依內容篩選」或「依關鍵字搜尋」，然後在「內容」下選擇「狀態」
2. 在「運算子」下選擇「狀態 =」。
3. 選擇「狀態 = 失敗」。
4. (選擇性) 選擇「清除篩選」以移除套用的篩選器。

事件視圖-過濾結果

根據屬性 (例如「類型」或日期和時間範圍) 套用一或多個篩選器，以限制顯示的資訊量。例如，若要檢視執行失敗的Task狀態步驟，請套用下列篩選器：

1. 選擇「依內容篩選」或「依關鍵字搜尋」，然後在「內容」下選擇「類型」
2. 在「運算子」下選擇「類型 =」。
3. 選擇「類型 =」 TaskFailed。
4. (選擇性) 選擇「清除篩選」以移除套用的篩選器。

事件視圖-檢查TaskFailed事件詳細信息

選擇TaskFailed事件 ID

旁邊的，以檢查其詳細資訊，包括出現在下拉式方塊中的輸入、輸出和資源叫用。

Redriving處決

您可以使redrive用重新啟動過去 14 天內未成功完成的標準工作流程的執行。其中包括失敗、中止或逾時的執行。

當您執redrive行時，它會從失敗的步驟繼續失敗的執行，並使用相同的輸入。 Step Functions保留成功步驟的結果和執行歷程記錄，而這些步驟在您執行時不會重新redrive執行。例如，假設您的工作流程包含兩種狀態：一個Pass狀態後跟一個任務狀態。如果您的工作流程執行在「任務」狀態失敗，且您執redrive行，則執行會重新排程，然後重新執行「任務」狀態。

Redriven執行使用與原始執行嘗試相同的狀態機器定義和執行 ARN。如果您的原始執行嘗試與版本、別名或兩者相關聯，則redriven執行會與相同的版本、別名或兩者相關聯。即使您將別名更新為指

向不同版本，redriven執行仍會繼續使用與原始執行嘗試相關聯的版本。由於redriven執行使用相同的狀態機器定義，因此如果您更新狀態機器定義，則必須開始新的執行。

當您執redrive行時，狀態機器層級逾時 (如果已定義) 會重設為 0。如需狀態機器層級逾時的詳細資訊，請參閱`TimeoutSeconds`。

執行redrives被認為是狀態轉換。有關狀態轉換如何影響計費的詳細資訊，請參閱 Step Functions 定價。

主題

Redrive未成功執行的資格

如果您的原始執行嘗試符合以redrive下條件，則可以執行：

- 您在 2023 年 11 月 15 日或之後開始執行。您在此日期之前開始的執行不符合redrive資格。
- 執行狀態不是`SUCCEEDED`。
- 工作流程執行未超過 14 天的redrivable期限。 Redrivable期間是指在此期間，你可以給定redrive的執行的時間。此期間從狀態機完成其執行之日開始。
- 工作流程執行未超過一年的開啟時間上限。如需狀態機器執行配額的相關資訊，請參閱與狀態機器執行相關的配額。
- 執行事件歷史記錄計數小於 24,999。 Redriven執行追加他們的事件歷史記錄到現有的事件歷史記錄。請確定您的工作流程執行包含少於 24,999 個事件，以容納`ExecutionRedriven`歷史記錄事件和至少一個其他歷史記錄事件。

Redrive個別狀態的行為

視工作流程中失敗的狀態而定，所有失敗狀態的redrive行為會有所不同。下表說明所有狀態的redrive行為。

州名	Redrive執行行為
Pass	如果前面的步驟失敗或狀態機器逾時，「通過」狀態就會結束，且不會在上redrive執行。
任務	排程並再次啟動工作狀態。 當您重新redrive執行「TimeoutSeconds 工作」狀態的執行時，如果已定義的狀態，則會重設為 0。如需逾時的詳細資訊，請參閱工作狀態。
Choice	重新評估「選擇」狀態規則。
等候	如果狀態指TimestampPath 定Timestamp 或參照過去的時間戳記，則redrive會導致「等待」狀態退出並進入Next欄位中指定的狀態。
Succeed	不會聲明進入「成功」redrive 狀態的機器執行。
Fail	重新進入「失敗」狀態，然後再次失敗。
平行	重新排程，並redrives僅重新排程失敗或中止的分支。 如果狀態因為States.DataLimitExceeded 錯誤而失敗，則會重新執行「平行」狀態，包括在原始執行嘗試中成功的分支。
內聯映射狀態	重新排程並redrives僅重新排程失敗或中止的版序。 如果狀態因為States.DataLimitExceeded 錯誤而失敗，則會重新執行 Inline Map 狀態，包括在原始執行嘗試中成功的反覆項目。
分散式地圖狀態	redrivesMap R un 中不成功的子工作流程執行。如需詳細資訊，請參閱 Redriving地圖運行。

州名	Redrive執行行為
	如果狀態因為States.DataLimitExceeded 錯誤而失敗，則會重新執行「分散式貼圖」狀態。這包括在原始執行嘗試中成功的子工作流程。

執行的 redrive IAM 許可

Step Functions 需要適當的權限redrive來執行。以下 IAM 政策示例授予狀態機執行所需的最低權限。redriving請記住將*##*文本替換為特定於資源的信息。

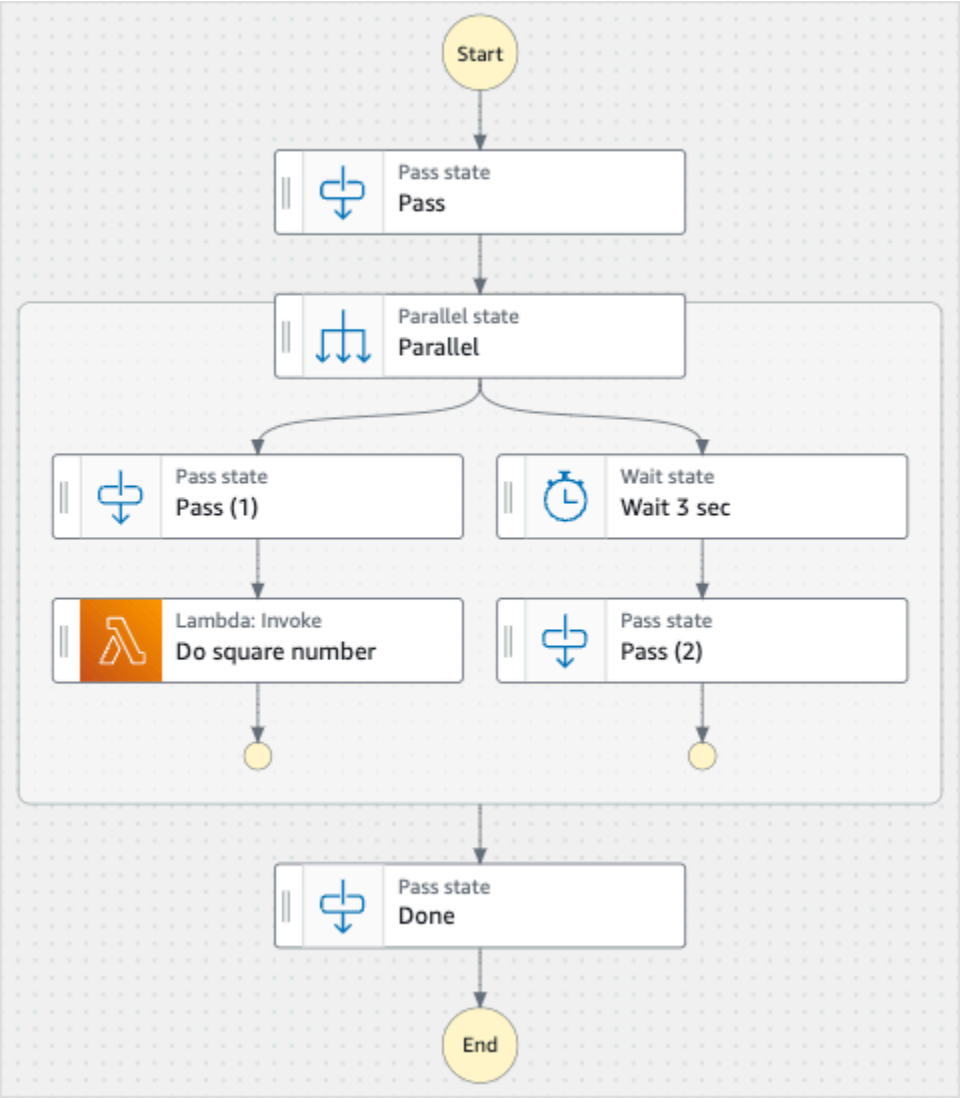
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

如需 Map Run 所需權限的範例，請參閱分散式地圖的 IAM 政策範例 redriving。redrive

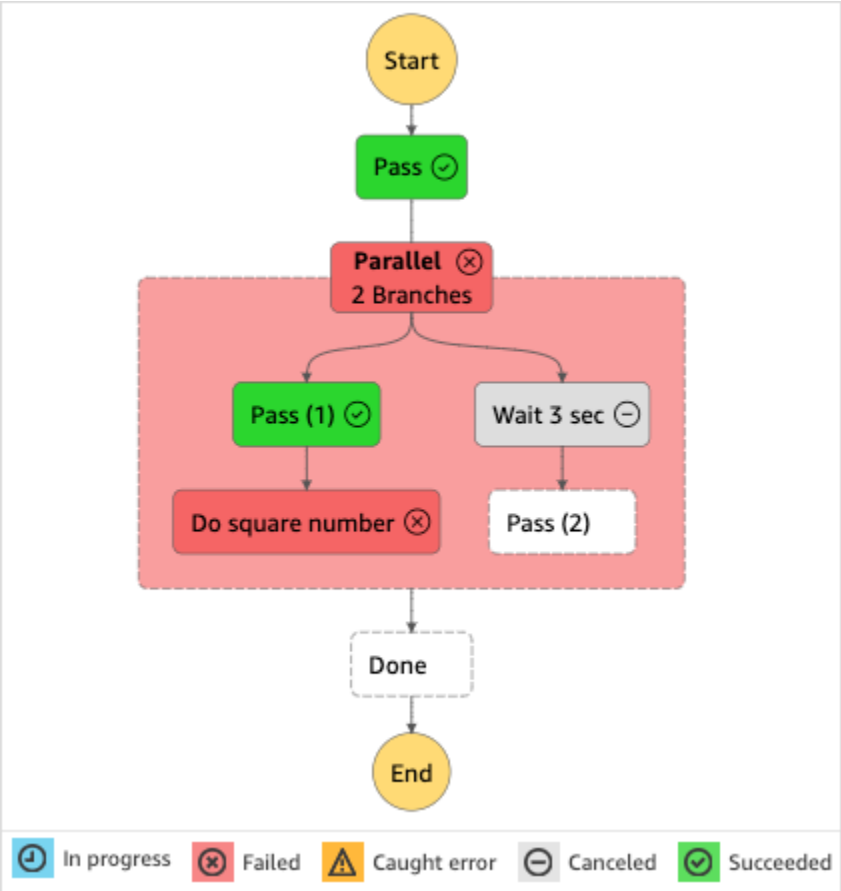
Redriving控制台中的執行

您可以從Step Functions控制台進行redrive合格的執行。

例如，假設下列影像代表狀態機的工作流程圖形。



想像一下你運行這個狀態機器。下圖顯示狀態機執行的圖形。



如此影像所示，「平行」狀態內名為 D o 平方數的「Lambda叫用」步驟傳回錯誤。這會造成「平行」狀態失敗。執行中或未啟動的分支會停止，且狀態機器執行失敗。

從redrive控制台執行

1. 開啟 Step Functions 主控台，然後選擇執行失敗的現有狀態機器。
2. 在狀態機器詳細資料頁面的 [執行項目] 下，選擇失敗的執行執行個體。
3. 選擇 Redrive。
4. 在Redrive對話方塊中，選擇Redrive執行。

> **Tip**
>
> 如果您位於失敗執行的 [執行詳細資訊] 頁面上，請執行下列其redrive中一項動作：
>
> - 選擇 [復原]，然後Redrive從失敗中選取。
> - 選擇「動作」，然後選取Redrive。

請注意，redrive使用相同的狀態機器定義和 ARN。它會繼續從原始執行嘗試中失敗的步驟執行。在此範例中，它是「平行」狀態內的「做平方數」步驟和「等待 3 秒」分支。在「平行」狀態下重新啟動這些失敗的步驟執行之後，redrive會繼續執行「完成」步驟。

5. 選擇執行項目以開啟「執行項目詳細資訊」頁面

   您可以在此頁面檢視redriven執行結果。例如，在該執行摘要部分中，您可以看到 Redrivecount，它代表執行的次數redriven。在「事件」區段中，您可以看到附加至原始執行嘗試事件的redrive相關執行事件。例如，`ExecutionRedriven`事件。

Redriving使用 API 執行

您可以使用 RedriveExecutionAPI 進行redrive合格的執行。此 API 會從失敗、中止或逾時的步驟重新啟動標準工作流程的失敗執行。

在 AWS Command Line Interface (AWS CLI) 中，執行下列命令以執行不redrive成功的狀態機器。請記住將*##*文本替換為特定於資源的信息。

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

檢查redriven執行

您可以在主控台或使用 API：GetExecutionHistory和檢查redriven執行DescribeExecution。

檢查控redriven制台上的執行

1. 開啟 Step Functions 主控台，然後選擇您已redriven執行的現有狀態機器。
2. 開啟「執行詳細資訊」頁面。

   您可以在此頁面檢視redriven執行結果。例如，在該執行摘要部分中，您可以看到 Redrivecount，它代表執行的次數redriven。在「事件」區段中，您可以看到附加至原始執行嘗試事件的redrive相關執行事件。例如，`ExecutionRedriven`事件。

使用 redriven API 檢查執行

如果您已執redriven行狀態機器，則可以使用下列其中一個 API 來檢視有關redriven執行的詳細資料。請記住將*##*文本替換為特定於資源的信息。

- GetExecutionHistory — 以事件清單形式傳回指定執行的歷史記錄。此 API 也會傳回有關執行redrive嘗試的詳細資訊 (如果有的話)。

  在中 AWS CLI，執行下列命令。

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution — 提供有關狀態機器執行的資訊。這可以是與執行、執行輸入和輸出、執行redrive詳細資訊 (如果有的話) 以及相關執行中繼資料相關聯的狀態機器。

  在中 AWS CLI，執行下列命令。

  ```
  aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

redriven執行的重試行為

如果您的redriven執行重新執行已定義重試的任務平行、或內嵌對應狀態，則這些狀態的重試嘗試次數會重設為 0。這允許嘗試的最大次數redrive。對於redriven執行，您可以使用主控台追蹤這些狀態的個別重試嘗試。

檢查主控台中的個別重試嘗試

1. 在 Step Functions 主控台的 [執行詳細資訊] 頁面上redrive，選擇重試的狀態。
2. 選擇重試和redrives標籤。
3. 選擇每個重試嘗試的▶下一個，以檢視其詳細資料。如果重試嘗試成功，您可以在「輸出」(Output) 中檢視出現在下拉式方塊中的結果。

下列影像顯示針對原始執行嘗試和該執行的狀態所執行redrives的重試範例。在此影像中，會在原始嘗試和執行嘗試中redrive執行三次重試。第四次redrive嘗試執行成功，並傳回 16 的輸出。

Input | Output | Details | Definition | **Retries & redrives** | Events

Type	Status	Resource	Duration	Time
▶ Original execution	Failed	Logs | Lambda | Log group	00:00:00.151	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.139	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.164	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.149	
▶ Redrive #1	Failed	Logs | Lambda | Log group	00:00:00.187	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.147	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.154	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.170	
▶ Redrive #2	Failed	Logs | Lambda | Log group	00:00:00.206	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.184	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.188	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.219	
▶ Redrive #3	Failed	Logs | Lambda | Log group	00:00:00.198	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.142	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.174	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.208	
▼ Redrive #4	Succeeded	Logs | Lambda | Log group	00:00:00.195	

Output Learn more

```
1 ▾ {
2     "Squared": 16
3 }
```

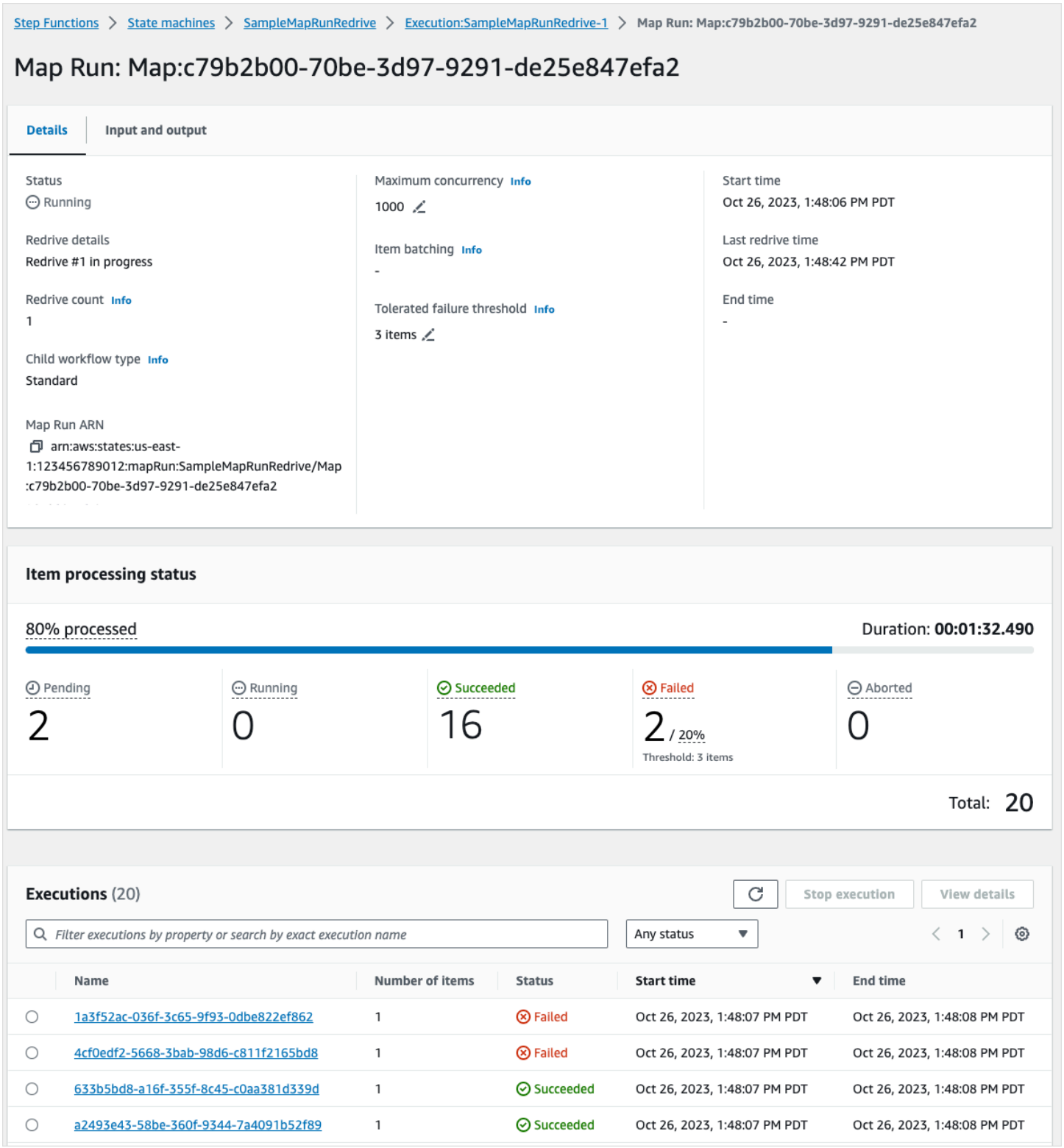
Formatted

檢查分散式地圖狀態執行的對應執行

當您在分散式模式下執行Map狀態時，「Step Functions」會建立「對應執行」資源。Map Run 是指分散式地圖狀態啟動的一組子工作流程執行，以及控制這些執行的執行階段設定。Step Functions 將 Amazon 資源名稱（ARN）分配給您的地圖運行。您可以在 Step Functions 控制台中檢查地圖運行。您也可以叫用 `DescribeMapRun` API 動作。「地圖執行」也會向其發出 CloudWatch度量。

「Step Functions」主控台會提供「對應執行詳細資訊」頁面，此頁面會顯示與「分散式對應」狀態執行相關的所 例如，您可以檢視「分散式對應」狀態的執行狀態、Map Run 的 ARN，以及由「分散式對映」狀態開始的子工作流程執行中所處理的項目狀態。您也可以檢視所有子工作流程執行的清單，並存取其詳細資訊。此外，如果您的地圖運行是 redriven，則可以在部分中查看地圖運行的redrive映射運行執行摘要詳細信息。例如，最後一redrive次。控制台會以儀表板格式顯示此資訊。

「對應執行詳細資訊」頁面包含下列段落：



目錄

- 料號處理狀態
- 執行清單
- Redriving地圖運行
  - Redrive在 Map Run 中使用子工作流程的資格
  - 子工作流程執redrive行行為
  - 地圖運行中使用的輸入場景 redrive
  - 對地圖運行redrive的 IAM 許可
  - Redriving在控制台中運行地圖
  - Redriving使用 API 執行地圖

映射運行執行摘要

「對應執行執行項目摘要」段落會顯示在「對應執行詳細資訊」頁面頂端。本節提供「分散式對應」狀態之執行詳細資訊的概觀。此資訊會在下列標籤之間劃分：

詳細資訊

顯示資訊，例如「分散式對應」狀態的執行狀態、Map 執行 ARN，以及由「分散式對應」狀態開始的子工作流程執行類型。您可以檢視其他組態，例如「對映執行」的容許失敗臨界值，以及為子工作流程執行指定的最大並行處理。您也可以編輯這些模型組態。

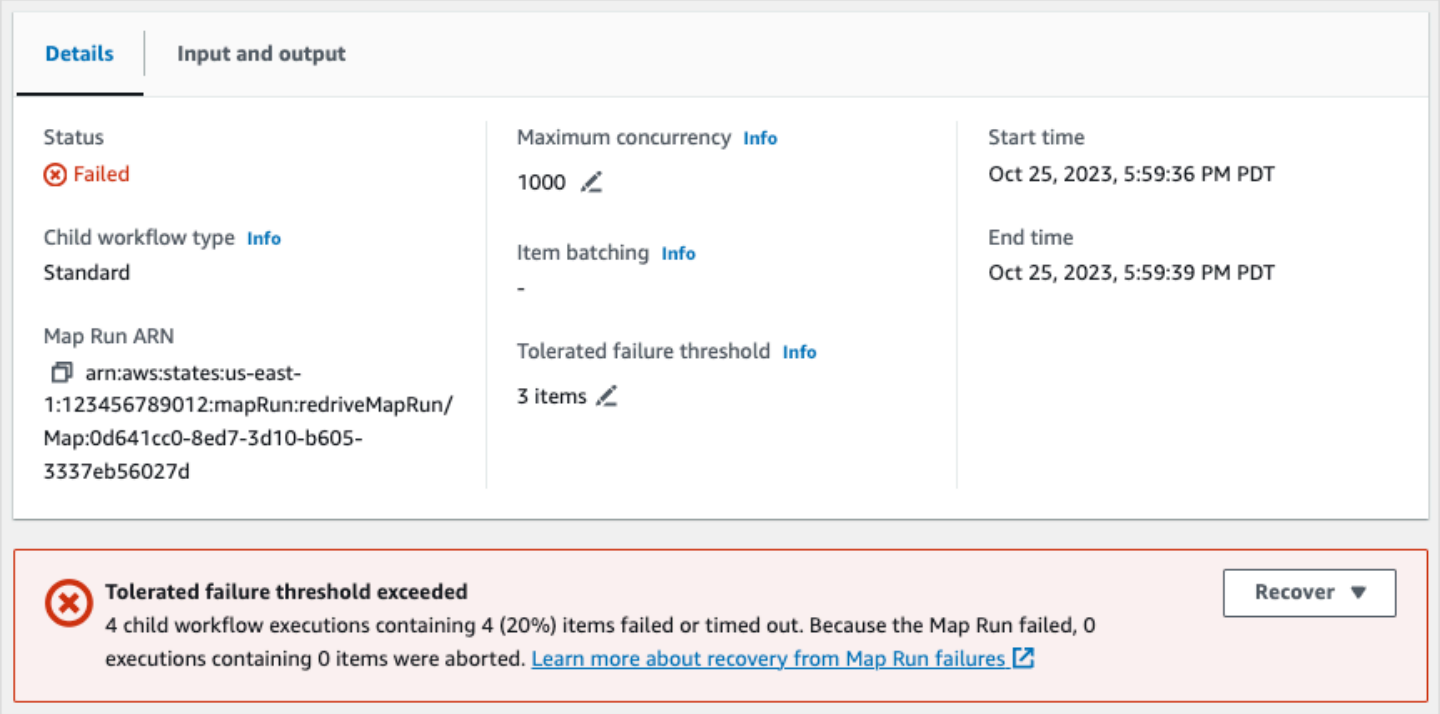
輸入和輸出

顯示分散式地圖狀態接收的輸入及其產生的對應輸出。例如，您可以檢視輸入資料集及其位置，以及套用至該資料集中個別資料項目的輸入篩選器。如果您匯出分散式地圖狀態執行的輸出，此索引標籤會顯示包含執行結果的 Amazon S3 儲存貯體的路徑。否則，它會將您指向父工作流程的「執行詳細資訊」頁面，以檢視執行輸出。

錯誤訊息

如果您的 Map 執行失敗，[對應執行詳細資訊] 頁面會顯示錯誤訊息，說明失敗的原因。

從此錯誤訊息上的 [復原] 下拉式按鈕中，您可以redrive使用此 Map Run 啟動的失敗子工作流程執行，也可以啟動父工作流程的新執行。如需詳細資訊，請參閱 Redriving地圖運行。



料號處理狀態

「料號處理狀態」區段會顯示「對映執行」中處理之料號的狀態。例如，「待處理」 表示子工作流程執行尚未開始處理項目。

項目狀態取決於處理項目的子工作流程執行的狀態。如果子工作流程執行失敗、逾時或使用者取消執行，Step Functions 就不會收到有關該子工作流程執行內之項目處理結果的任何資訊。該執行處理的所有項目都會共用子工作流程執行的狀態。

例如，假設您要在兩個子工作流程執行中處理 100 個項目，其中每個執行都會處理 50 個項目的批次。如果其中一個執行失敗而另一個執行成功，則您將擁有 50 個成功和 50 個失敗的項目。

下表說明所有料號可用的處理狀態型態：

狀態	描述
待定	指出子工作流程執行尚未開始處理的項目。如果 Map Run 停止、失敗或使用者在處理項目開始之前取消執行，則該項目將保持在「待處理」狀態。

狀態	描述
	例如，如果 Map Run 失敗且有 10 個待處理的項目，則這 10 個項目仍處於「擱置中」狀態。
執行中	指出子工作流程執行目前正在處理的項目。
成功	指出子工作流程執行已成功處理項目。 成功的子工作流程執行不能有任何失敗的項目。如果資料集中的某個項目在執行期間失敗，則整個子工作流程執行都會失敗。
失敗	指出子工作流程執行無法處理項目，或執行逾時。如果子工作流程執行處理的任何一個項目失敗，則整個子工作流程執行都會失敗。 例如，考慮一個處理 1000 個項目的子工作流程執行。如果該資料集中的任何一個項目在執行期間失敗，則 Step Functions 會將整個子工作流程執行視為失敗。 當您執redrive行 Map Run 時，具有此狀態的項目計數會重設為 0。

狀態	描述
已中止	指出子項工作流程執行已開始處理項目，但使用者已取消執行，或是「Step Functions」因為「對應執行」失敗而停止執行。 例如，請考慮正在處理 50 個項目的執行子工作流程執行。如果 Map Run 因為失敗或使用者取消了執行而停止，則子工作流程執行和所有 50 個項目的狀態都會變更為「已中止」。 如果您使用 Express 類型的子工作流程執行，則無法停止執行。 當您redrive啟動 Express 類型的子工作流程執行的 Map Run 時，具有此狀態的項目計數將重設為 0。這是因為 Express 子工作流程會使用 StartExecutionAPI 動作而不是重新啟動 redriven。

執行清單

「執行項目」段落會列出特定 Map Run 的所有子工作流程執行項目。使用 [按確切執行名稱搜尋] 欄位來搜尋特定的子工作流程執行項目。您也可以使用「任何狀態」 下拉式清單，依狀態篩選子工作流程執行歷史記錄。若要查看有關特定執行項目的詳細資訊，請從清單中選取子項工作流程執行，然後選擇檢視詳細資訊按鈕以開啟其執行詳細資訊頁面。

> **Important**
>
> 子工作流程執行的保留原則為 90 天。超過此保留期間的已完成子工作流程執行不會顯示在「執行項目」表格中。即使「分散式對應」狀態或父工作流程的執行時間長於保留期間，也是如此。如果您使`ResultWriter`用將分散式地圖狀態輸出匯出到 Amazon S3 儲存貯體，則可以檢視這些子工作流程執行的執行詳細資料 (包括結果)。

> **Tip**
>
> 選擇重新整理按
>
> 鈕 
>
> 檢視所有子工作流程執行的最新清單。

以

Redriving地圖運行

您可以在父工作流程的 Map Run 中重新啟動不成功redriving的子工作流程執行。redriven父工作流程redrives所有不成功的狀態，包括分佈式地圖。父工作流程在父工作流程完成其執行時，如果沒<stateType>Exited有與狀態相對應的<stateType>Entered事件相對應的事件，父工作流程會重新驅動失敗狀態。例如，如果事件歷史記錄不包含MapStateExited事件的事件，則可以redrive將父工作流程用於 Map Run 中redrive所有不成功的子工作流程執行。MapStateEntered

當狀態機器沒有存取、或兩者所需的權限時，在原始執行嘗試中 ItemReaderResultWriter，Map Run 不會啟動或失敗。如果未在父工作流程的原始執行嘗試中啟動 Map Run，redriving則父工作流程首次啟動 Map Run。若要保留此功能，請將所需的權限新增至您的狀態機器角色，然後再新增父工作流redrive程。如果您redrive的父工作流程沒有添加所需的權限，它會嘗試啟動新的 Map Run 運行，該運行將再次失敗。如需有關您可能需要的權限的資訊，請參閱使用分散式地圖狀態的 IAM 政策。

主題

- Redrive在 Map Run 中使用子工作流程的資格
- 子工作流程執redrive行行為
- 地圖運行中使用的輸入場景 redrive
- 對地圖運行redrive的 IAM 許可
- Redriving在控制台中運行地圖
- Redriving使用 API 執行地圖

Redrive在 Map Run 中使用子工作流程的資格

如果滿足以redrive下條件，則可以在 Map Run 中執行失敗的子工作流程執行：

- 您在 2023 年 11 月 15 日或之後開始執行父工作流程。您在此日期之前開始的執行不符合redrive資格。

- 你還沒有超過給定的地圖運行 redrives 1000 的硬性限制。如果您超過此限制，就會收到`States.Runtime`錯誤訊息。
- 父工作流程為redrivable。如果父工作流程不是redrivable，則無法redrive在 Map Run 中執行子工作流程。如需有關工作流程redrive資格的詳細資訊，請參閱Redrive未成功執行的資格。
- Map Run 中「標準」類型的子工作流程執行未超過 25,000 執行事件歷史記錄限制。超過事件歷史記錄限制的子工作流程執行會計入容忍的失敗臨界值，並視為失敗。如需執行redrive資格的詳細資訊，請參閱Redrive未成功執行的資格。

即使在原始執行嘗試中的 Map Run 失敗，redriven在以下情況下，仍會啟動新的 Map Run，並且現有的 Map Run 不會出現：

- 由於錯誤，地圖運行失`States.DataLimitExceeded`敗。
- 地圖運行失敗，因為 JSON 數據插值錯誤，`States.Runtime`. 例如，您在中選取了一個不存在的 JSON 節點。OutputPath

即使父工作流程停止或逾時，「地圖執行」仍可繼續執行。在這些情況下，redrive不會立即發生：

- Map Run 可能仍會取消進行中類型為標準的子工作流程執行，或等待 Express 類型的子工作流程執行完成其執行。
- 如果您將 Map Run 設定為匯出結果ResultWriter，則可能仍會將結果寫入。

在這些情況下，執行中的 Map Run 會先完成其作業，然後再嘗試執行redrive。

子工作流程執redrive行行為

Map Run 中的redriven子工作流程執行會顯示如下表所述的行為。

快速子工作流程	標準子工作流程
在原始執行嘗試中失敗或逾時的所有子工作流程執行都會使用 StartExecutionAPI 動作啟動。中的第一個狀態ItemProcessor會先執行。	在原始執行嘗試中失敗、逾時或取消的所有子工作流程執行都redriven使用 RedriveExecutionAPI 動作。這些子工作流程redriven來自上一個狀態 ItemProcessor ，導致其執行失敗。
不成功的執行總是可以。redriven這是因為 Express 子工作流程執行始終會使用 StartExecution API 動作作為新執行來啟動。	不成功的標準子工作流程執行不能redriven總是如此。如果執行不是redrivable，它將不會再次嘗試。執行的最後一個錯誤或輸出是永久的。當

快速子工作流程	標準子工作流程
	執行超過 25,000 個歷史記錄事件或其 14 天的期限已過redrivable期時，這是可能的。 redrivable如果父工作流程執行在 14 天內關閉，但子工作流程執行早於 14 天關閉，則「標準」子工作流程執行可能不是。
Express 子工作流程執行使用與原始執行嘗試相同的執行 ARN，但您無法清楚地識別它們的個人。redrives	標準子工作流程執行使用與原始執行嘗試相同的執行 ARN。您可以在控制台redrives中清楚地識別個人並使用 API，例如GetExecutionHistory和DescribeExecution。如需詳細資訊，請參閱檢查redriven執行。

如果您redriven有 Map Run，並且已達到並發限制，則該 Map Run 中的子工作流程執行將轉換為待處理狀態。Map Run 的執行狀態也會轉換為「擱置中」redrive 狀態。在指定的並行限制可以允許執行更多子工作流程執行之前，執行仍處於「擱置redrive中」狀態。

例如，假設工作流程中分散式地圖的並行限制為 3000，而要重新執行的子工作流程數為 6000。這會導致 3000 個子工作流程並行執 parallel，而剩餘 3000 個工作流程仍處於擱置中的重新磁碟機狀態。在第一批 3000 個子工作流程完成其執行之後，系統會執行剩餘的 3000 個子工作流程。

當 Map Run 完成執行或中止時，處於「擱置中」redrive 狀態的子工作流程執行計數會重設為 0。

地圖運行中使用的輸入場景 redrive

根據您在原始執行嘗試中向分散式地圖提供輸入的方式而定，redrivenMap Run 將使用輸入，如下表所述。

原始執行嘗試中的輸入	在地圖運行中使用的輸入 redrive
從先前的狀態或執行輸入傳遞的輸入。	redriven地圖運行使用相同的輸入。
使用傳遞的輸入ItemReader和 Map Run 未啟動子工作流程執行，因為下列其中一個條件成立： • 地圖運行失敗，`States.ItemReaderFailed` 錯誤。	地redriven圖運行使用 Amazon S3 存儲桶中的輸入。

原始執行嘗試中的輸入	在地圖運行中使用的輸入 redrive
• 地圖運行失敗，States.ResultWriterFailed 錯誤。 • 在「Map Run」開始之前，父工作流程執行已逾時或取消。	
輸入通過使用 ItemReader. 啟動或嘗試啟動子工作流程執行後，Map 執行失敗。	redrivenMap Run 使用原始執行嘗試中提供的相同輸入。

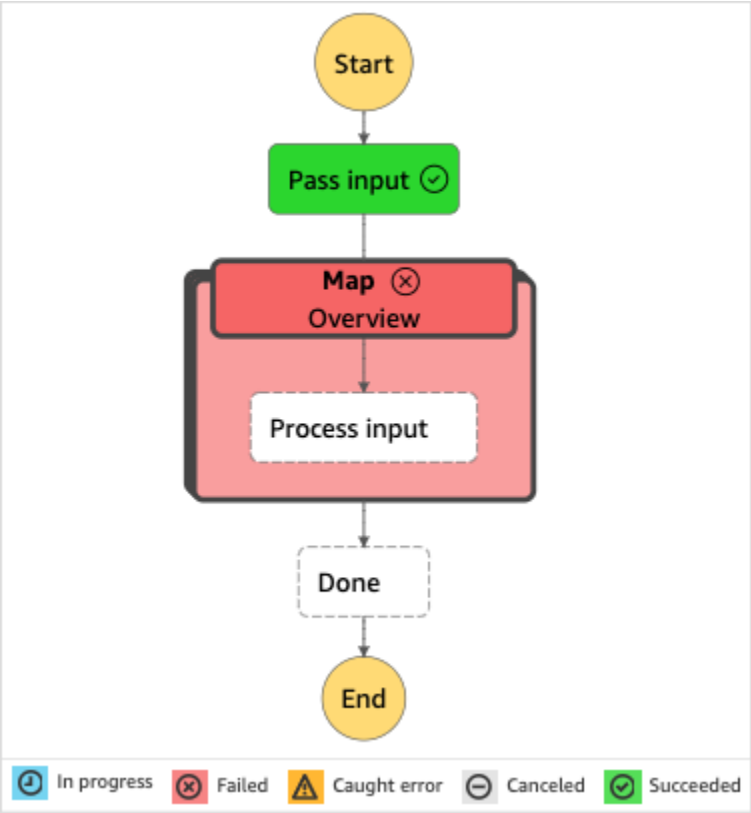
對地圖運行redrive的 IAM 許可

Step Functions 需要對映執行redrive的適當權限。以下 IAM 政策示例授予狀態機器進行 Map Run 所需的最低權限。redriving請記住將##文本替換為特定於資源的信息。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

Redriving在控制台中運行地圖

下圖顯示了包含分散式映射的狀態機器的執行圖形。此執行失敗，因為映射運行失敗。對於redrive「地圖執行」，您必須使redrive用父工作流程。



從控制台到redrive地圖運行

1. 開啟 Step Functions 主控台，然後選擇包含執行失敗的分散式對應的現有狀態機器。
2. 在狀態機器詳細資料頁面的 Executions 下，選擇此狀態機器的失敗執行個體。
3. 選擇 Redrive。
4. 在Redrive對話方塊中，選擇Redrive執行。

> **Tip**
>
> 您也redrive可以從「執行詳細資訊」或「對映執行詳細資訊」頁面執行對應執行。
> 如果您在 [執行詳細資訊] 頁面上，請執行下列其redrive中一項動作：
>
> - 選擇 [復原]，然後Redrive從失敗中選取。
> - 選擇「動作」，然後選取Redrive。
>
> 如果您在 [對應執行詳細資料] 頁面上，請選擇 [復原]，然後Redrive從失敗中選取。

請注意，redrive使用相同的狀態機器定義和 ARN。它會從原始執行嘗試中失敗的步驟繼續執行執行。在此範例中，它是名為 Map 的分散式對應步驟，並在其中的處理程序輸入步驟。重新啟動「Map Run」不成功的子工作流程執行後，redrive將繼續執行「完成」步驟。

5. 在「執行詳細資訊」頁面中，選擇「對映執行」以查看「對redriven映執行」的詳細資訊。

   您可以在此頁面檢視redriven執行結果。例如，在該映射運行執行摘要部分中，您可以看到Redrivecount，它表示地圖運行已經的次數redriven。在「事件」區段中，您可以看到附加至原始執行嘗試事件的redrive相關執行事件。例如，`MapRunRedriven`事件。

完成 Map Run 之後，您可以在主控台或使用GetExecutionHistory和 DescribeExecutionAPI 動作檢查其redrive詳細資料。redriven如需有關檢查redriven執行的詳細資訊，請參閱檢查redriven執行。

Redriving使用 API 執行地圖

您可以redrive使用父工作流程上的 RedriveExecutionAPI 執行合格的 Map Run。此 API 會在 Map Run 中重新啟動不成功的子工作流程執行。

在 AWS Command Line Interface (AWS CLI) 中，執行下列命令以執行不redrive成功的狀態機器。請記住將*##*文本替換為特定於資源的信息。

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

完成 Map Run 後，您可以在主控台或使用 DescribeMapRunAPI 動作檢查其redrive詳細資料。redriven若要檢查「對映執行」中標準工作流程執行的redrive詳細資訊，您可以使用GetExecutionHistory或 DescribeExecutionAPI 動作。如需有關檢查redriven執行的詳細資訊，請參閱檢查redriven執行。

如果您已在父工作流程上啟用記錄功能，則可以在 Step Functions 主控台上的 Map Run 中檢查 Express 工作流程執行的redrive詳細資料。如需更多詳細資訊，請參閱 記錄使用CloudWatch日誌。

Step Functions 中的錯誤處理

除了`Pass`和狀態之外的所有`Wait`狀態都可能會遇到執行階段錯誤。錯誤可能由於各種原因而發生，例如以下示例：

- 狀態機器定義問題 (例如，`Choice` 狀態中沒有相符的規則)

- 任務失敗 (例如，AWS Lambda 函數中的例外狀況)
- 暫時性問題 (例如，網路分割區事件)

根據預設，當狀態回報錯誤時，AWS Step Functions 會導致完全無法執行。

> **Tip**
>
> 若要部署包含錯誤處理的工作流程範例AWS 帳戶，請參閱單元 8-AWS Step Functions工作坊的錯誤處理。

主題

錯誤名稱

Step Functions 使用區分大小寫的字串 (稱為錯誤名稱) 識別 Amazon 州語言中的錯誤。Amazon States 語言會定義一組內建字串，這些字串命名為已知錯誤，全部都以前置`States.`詞開頭。

`States.ALL`

符合任何已知錯誤名稱的萬用字元。

> **Note**
>
> 此錯誤類型無法 catch `States.DataLimitExceeded` 終端錯誤類型和運行時錯誤類型。如需這些錯誤類型的詳細資訊，請參閱`States.DataLimitExceeded`和`States.Runtime`。

`States.DataLimitExceeded`

「Step Functions」會在下列情況下報告`States.DataLimitExceeded`例外：

- 當連接器的輸出大於有效負載大小配額時。
- 當狀態的輸出大於有效負載大小配額時。
- Parameters處理之後，狀態的輸入大於有效負載大小配額時。

如需配額的詳細資訊，請參閱配額。

> **Note**
>
> 這是錯誤類型無法捕獲的終端States.ALL錯誤。

States.ExceedToleratedFailureThreshold

Map狀態失敗，因為失敗的項目數超過狀態機器定義中指定的臨界值。如需詳細資訊，請參閱分散式對應狀態的容許失敗臨界值。

States.HeartbeatTimeout

狀Task態無法傳送活動訊號超過該HeartbeatSeconds值的時間。

> **Note**
>
> 此錯誤僅在Catch和Retry欄位中可用。

States.IntrinsicFailure

嘗試在有效負載範本中叫用內建函式失敗。

States.ItemReaderFailed

狀Map態失敗，因為無法從ItemReader欄位中指定的項目來源讀取。如需詳細資訊，請參閱ItemReader。

States.NoChoiceMatched

Choice狀態無法將輸入與「選擇規則」中定義的條件相符，且未指定「預設」轉移。

States.ParameterPathFailure

嘗試取代狀態欄位中名稱以.$使用路徑結尾的Parameters欄位失敗。

States.Permissions

Task狀態失敗，因為它沒有足夠的權限來運行指定的代碼。

States.ResultPathMatchFailure

Step Functions 無法將狀態的ResultPath字段應用於輸入接收到的狀態。

States.ResultWriterFailed

狀Map態失敗，因為無法將結果寫入ResultWriter欄位中指定的目的地。如需詳細資訊，請參閱ResultWriter。

States.Runtime

由於某些無法處理的例外狀況，執行失敗。這些通常是在運行時間的錯誤所導致，例如嘗試在 Null JSON 承載上套用 InputPath 或 OutputPath。States.Runtime錯誤無法重新擷取，而且永遠會導致執行失敗。重試或 catch States.ALL 獲不會 catch 獲States.Runtime錯誤。

States.TaskFailed

在執行期間失敗的 Task 狀態。在重試或 catch 中使用時，States.TaskFailed充當通配符，匹配除了任何已知的錯誤名稱States.Timeout。

States.Timeout

執行時間超過 TimeoutSeconds 值，或無法傳送活動訊號的時間超過 HeartbeatSeconds 值的 Task 狀態。

此外，如果狀態機器的執行時間超過指定TimeoutSeconds值，則執行會失敗並顯示States.Timeout錯誤。

狀態可以回報具有其他名稱的錯誤。但是，這些錯誤名稱不能以States.前綴開頭。

為確保最佳實務，請確認生產程式碼可以處理 AWS Lambda 服務例外狀況 (Lambda.ServiceException 和 Lambda.SdkClientException)。如需詳細資訊，請參閱處理 Lambda 務例外。

> **Note**
>
> Lambda 中未處理的錯誤會報告為錯誤輸出Lambda.Unknown中。這些包括 out-of-memory 錯誤和功能超時。您可以在Lambda.UnknownStates.ALL、或上進行比對States.TaskFailed來處理這些錯誤。當 Lambda 達到調用的最大數量時，錯誤

為。Lambda.TooManyRequestsException如需 Lambda 函數錯誤的詳細資訊，請參閱AWS Lambda開發人員指南中的錯誤處理和自動重試。

發生錯誤後重試

Task、Parallel、和Map狀態可以有一個名為的欄位Retry，其值必須是稱為擷取器的物件陣列。個別的 Retrier 代表特定的重試次數，通常為較長的時間間隔。

當其中一個狀態報告錯誤且有Retry欄位時，Step Functions 會依陣列中列出的順序掃描擷取器。當錯誤名稱出現在檢索器欄位的值中時，狀態機會按照ErrorEquals欄位中的定義進行重試嘗試。Retry

如果您的redriven執行重新執行已定義重試次數的任務平行、或 Inline Map 狀態，則這些狀態的重試嘗試次數會重設為 0，以允許嘗試次數上限。redrive對於redriven執行，您可以使用主控台追蹤這些狀態的個別重試嘗試。如需詳細資訊，請參閱 Redriving處決 中的 redriven執行的重試行為。

Retrier 包含下列欄位：

> **Note**
>
> 重試被視為狀態轉換。有關狀態轉換如何影響計費的詳細資訊，請參閱 Step Functions 定價。

`ErrorEquals` (必要)

符合錯誤名稱的非空白字串陣列。當狀態報告錯誤時，Step Functions 會透過擷取器進行掃描。當錯誤名稱出現於此陣列時，它會實作此 Retrier 中所述的重試政策。

`IntervalSeconds` (選用)

正整數；代表第一次嘗試重試前的秒數 (1預設情況下)。 IntervalSeconds其最大值為 99999999。

`MaxAttempts` (選用)

正整數，其代表重試次數上限 (預設值 3)。如果出現錯誤的次數超過指定次數，則重試會停止且一般錯誤處理會繼續執行。值0指定永遠不會重試錯誤。 MaxAttempts其最大值為 99999999。

`BackoffRate` (選用)

每次重試嘗試後，表示重試間隔的乘數IntervalSeconds會增加。依預設，BackoffRate值會增加2.0。

例如，假設你IntervalSeconds是 3，MaxAttempts是 3，BackoffRate是 2。第一次重試嘗試會在錯誤發生後三秒鐘進行。第二次重試會在第一次嘗試重試後六秒進行。第三次重試會在第二次重試嘗試後 12 秒進行。

MaxDelaySeconds (選用)

正整數；設定重試間隔可以增加的最大值 (以秒為單位)。此欄位對於欄位搭配使用很有幫助。BackoffRate您在此欄位中指定的值會限制套用至每次連續重試嘗試的輪詢率乘數所產生的指數等待時間。您必須指定一個大於 0 且小於 31622401 的值。MaxDelaySeconds

如果未指定此值，Step Functions 不會限制重試嘗試之間的等待時間。

JitterStrategy (選用)

字串，決定是否要在連續重試嘗試之間的等待時間中包含抖動。抖動會在隨機延遲間隔內分散這些嘗試，以減少同時重試的次數。該字符串接受FULL或NONE作為其值。預設值為 NONE。

例如，假設您已設定MaxAttempts為 3、IntervalSeconds為 2，並設定BackoffRate為 2。第一次重試嘗試會在錯誤發生後兩秒鐘進行。第二次重試會在第一次重試嘗試後四秒進行，第三次重試會在第二次重試嘗試後 8 秒進行。如果設定JitterStrategy為FULL，則第一個重試間隔會在 0 到 2 秒之間隨機排列，第二個重試間隔會在 0 到 4 秒之間隨機排列，第三個重試間隔會在 0 到 8 秒之間隨機排列。

重試欄位範例

本節包含下列Retry欄位範例。

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> 若要將錯誤處理工作流程的範例部署到您的AWS 帳戶，請參閱AWS Step Functions工作坊的錯誤處理模組。

範例 1 — 重試 BackoffRate

下列的範例會在等待三秒鐘後進行兩次重試嘗試，而第一次重試則會進行。Retry根據BackoffRate您指定的，「Step Functions 數」會增加每次重試之間的間隔，直到達到重試嘗試次數上限為止。在下列範例中，第二次重試嘗試會在第一次重試後等待三秒鐘後開始。

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
   "BackoffRate": 1
} ]
```

範例 2 — 重試 MaxDelaySeconds

下列範例會進行三次重試，並限制 5 秒後產生的BackoffRate等待時間。第一次重試會在等待三秒鐘後進行。第二次和第三次重試嘗試會在前一次重試嘗試後等待五秒鐘後發生，因為等待時間上限是由設定的最大等待時間限制MaxDelaySeconds。

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

如果沒有MaxDelaySeconds，則第二次重試會在第一次重試後六秒進行，第三次重試會在第二次重試後 12 秒進行。

範例 3 — 重試除狀態之外的所有錯誤。逾時

出現在 Retrier 的 ErrorEquals 欄位中的預留名稱 States.ALL 是符合任何錯誤名稱的萬用字元。它必須單獨顯示在 ErrorEquals 陣列中，而且必須顯示在 Retry 陣列的最後一個 Retrier 中。該名稱States.TaskFailed還具有通配符，並匹配除外的任何錯誤States.Timeout。

下列Retry欄位範例會重試除外States.Timeout的任何錯誤。

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
```

```
} ]
```

範例 4 — 複雜的重試案例

Retrier 的參數會套用於在單一狀態執行的範疇中對 Retrier 的所有造訪。

請考慮以下 Task 狀態。

```
"X": {
   "Type": "Task",
   "Resource": "arn:aws:states:us-east-1:123456789012:task:X",
   "Next": "Y",
   "Retry": [ {
      "ErrorEquals": [ "ErrorA", "ErrorB" ],
      "IntervalSeconds": 1,
      "BackoffRate": 2.0,
      "MaxAttempts": 2
   }, {
      "ErrorEquals": [ "ErrorC" ],
      "IntervalSeconds": 5
   } ],
   "Catch": [ {
      "ErrorEquals": [ "States.ALL" ],
      "Next": "Z"
   } ]
}
```

此工作連續失敗四次，輸出以下錯誤名稱：ErrorAErrorB、ErrorC、和ErrorB。因此會發生下列情況：

- 前兩個錯誤匹配第一個檢索器，並導致等待一秒和兩秒。
- 第三個錯誤匹配第二個檢索器，並導致等待五秒鐘。
- 第四個錯誤也匹配第一個檢索器。但是，對於該特定錯誤，它已經達到了兩次 retry（MaxAttempts）的最大值。因此，該擷取器會失敗，而且執行會透過Catch欄位將工作流程重新導向至Z狀態。

后援国家

TaskMap和Parallel狀態每個狀態都可以有一個名為的欄位Catch。此欄位的值必須物件陣列，也稱為 Catcher。

Catcher 包含下列欄位。

`ErrorEquals` (必要)

符合錯誤名稱的非空白字串陣列，其指定方式完全與同名的 Retrier 欄位一樣。

`Next` (必要)

字串，該字串必須完全符合其中一個狀態機器的狀態名稱。

`ResultPath` (選用)

決定捕捉器傳送到`Next`欄位中指定狀態的輸入的路徑。

當狀態報告錯誤且沒有`Retry`欄位，或者如果重試無法解決錯誤，Step Functions 會依陣列中列出的順序掃描擷取器。當錯誤名稱顯示於 Catcher 的 `ErrorEquals` 欄位值時，狀態機器會轉換到 `Next` 欄位中具名的狀態。

出現在 Catcher 的 `ErrorEquals` 欄位中的預留名稱 `States.ALL` 是符合任何錯誤名稱的萬用字元。它必須單獨顯示在 `ErrorEquals` 陣列中，而且必須顯示在 `Catch` 陣列的最後一個 Catcher 中。該名稱`States.TaskFailed`還具有通配符，並匹配除外的任何錯誤`States.Timeout`。

下列`Catch`欄位範例會轉換至 Lambda 函數輸出未處理的 Java 例外狀況`RecoveryState`時命名的狀態。否則，此欄位會轉換到 `EndState` 狀態。

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
   "Next": "EndState"
} ]
```

> **Note**
>
> 每個 Catcher 可以指定多個要處理的錯誤。

錯誤輸出

當 Step Functions 轉換到 catch 名稱中指定的狀態時，該對象通常包含該字段Cause。此欄位的值是人類可讀的錯誤描述。此物件也稱為「錯誤輸出」。

在這個範例中，第一個 Catcher 包含 ResultPath 欄位。其運作方式類似於狀態最上層中的 ResultPath 欄位，會造成兩種可能性：

- 它會取得該狀態執行的結果，並覆寫狀態輸入的全部或部分。
- 它會取得結果，並將結果新增至輸入。在捕手處理錯誤的情況下，狀態的執行結果是錯誤輸出。

因此，對於示例中的第一個捕獲器，捕手將錯誤輸出添加到輸入中，error-info如果輸入中還沒有具有此名稱的字段，則命名為的字段。然後，捕手將整個輸入發送到RecoveryState。對於第二個捕獲器，錯誤輸出覆蓋輸入，捕手僅將錯誤輸出發送到。EndState

> **Note**
>
> 如未指定 ResultPath 欄位，它會預設為 $，該值會選取並覆寫整個輸入。

當一個狀態同時具有Retry和Catch字段時，Step Functions 首先使用任何適當的檢索器。如果重試原則無法解決錯誤，「Step Functions」會套用相符的捕捉器轉換。

原因有效載荷和服務整合

捕手返回一個字符串有效載荷作為輸出。使用 Amazon Athena 等服務整合時AWS CodeBuild，您可能需要將Cause字串轉換為 JSON。以下具有內在函數的Pass狀態示例顯示瞭如何將Cause字符串轉換為 JSON。

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

使用重試和使用 Catch 的狀態機器示例

以下範例中定義的狀態機器假設存在兩個 Lambda 函數：一個永遠失敗，另一個等待足夠長的時間以允許狀態機器中定義的逾時發生。

這是一個 Node.js Lambda 函數的定義，該函數總是失敗，並返回消息`error`。在接下來的狀態機器範例中，會命名此 Lambda 函數`FailFunction`。如需建立 Lambda 函數的相關資訊，請參閱一步驟 1：建立 Lambda 函數節。

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

這是一個 Node.js Lambda 函數的定義，睡眠 10 秒。在接下來的狀態機器範例中，會命名此 Lambda 函數`sleep10`。

> **Note**
>
> 在 Lambda 主控台中建立此 Lambda 函數時，請記得將 [進階設定] 區段中的逾時值從 3 秒 (預設值) 變更為 11 秒。

```
exports.handler = (event, context, callback) => {
    setTimeout(function(){
    }, 11000);
};
```

使用「重試」處理失敗

此狀態機器會使用 `Retry` 欄位來重試失敗且輸出錯誤名稱 `HandledError` 的函數。它重試此函數兩次，並在重試之間進行指數輪詢。

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
```

```
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Retry": [ {
            "ErrorEquals": ["HandledError"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
      "End": true
      }
   }
}
```

此變體使用預先定義的錯誤代碼`States.TaskFailed`，該代碼與 Lambda 函數輸出的任何錯誤匹配。

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Retry": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
         "End": true
      }
   }
}
```

> **Note**
>
> 最佳實務是，參考 Lambda 函數的工作應該處理 Lambda 服務例外狀況。如需詳細資訊，請參閱處理 Lambda 務例外。

使用 Catch 處理失敗

此範例會使用 Catch 欄位。當 Lambda 函數輸出錯誤時，它會捕獲錯誤，並且狀態機轉換為狀fallback態。

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["HandledError"],
            "Next": "fallback"
         } ],
         "End": true
      },
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

此變體使用預先定義的錯誤代碼States.TaskFailed，該代碼與 Lambda 函數輸出的任何錯誤匹配。

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "Next": "fallback"
         } ],
      "End": true
```

```
      },
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

使用「重試」處理逾時

此狀態機器使用Retry欄位重試逾時的Task狀態，根據中TimeoutSeconds所指定的逾時值。Step Functions 數會在此Task狀態下重試 Lambda 函數叫用兩次，並在重試之間進行指數輪詢。

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:sleep10",
         "TimeoutSeconds": 2,
         "Retry": [ {
            "ErrorEquals": ["States.Timeout"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
         "End": true
      }
   }
}
```

使用 Catch 處理超時

此範例會使用 Catch 欄位。如果發生逾時，狀態機器就會轉換到 fallback 狀態。

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
```

```
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:sleep10",
         "TimeoutSeconds": 2,
         "Catch": [ {
            "ErrorEquals": ["States.Timeout"],
            "Next": "fallback"
         } ],
         "End": true
      },
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

> **Note**
>
> 您可以使用 ResultPath 來保留狀態輸入與錯誤。請參閱用於 ResultPath 將錯誤和輸入都包含在 Catch。

調用AWS Step Functions從其他服務

您可以設定數個其他服務來叫用狀態機器。基於狀態機工作流類型，您可以非同步或同步呼叫狀態機器。若要同步呼叫狀態機，請使用StartSyncExecutionAPI 呼叫或 Amazon API 閘道與快速工作流程整合。透過非同步叫用，Step Functions 會暫停工作流程執行，直到傳回工作 Token 為止。但是，等待任務 Token 確實會使工作流程同步。

您可以設定為叫用步驟函數的服務包括：

- AWS Lambda，使用StartExecution呼叫。
- Amazon API Gateway
- Amazon EventBridge
- AWS CodePipeline
- AWS IoT規則引擎

- AWS Step Functions

步驟函數調用由StartExecution配額。如需詳細資訊，請參閱：

- 配額

讀取步驟函數中的一致性

AWS Step Functions 中的狀態機器更新最後會保持一致。幾秒鐘內的所有StartExecution呼叫都將使用更新的定義和 roleArn (IAM 角色的 Amazon 資源名稱)。如果在呼叫 UpdateStateMachine 後立即開始執行，則可能會使用先前的狀態機器定義和 roleArn。

如需詳細資訊，請參閱下列內容：

- AWS Step Functions API 參考中的 UpdateStateMachine
- 更新中的工作流程AWS Step Functions 入門。

步驟函數中的標記

AWS Step Functions 支援狀態機器 (標準和快速) 和活動的標記。這可協助您追蹤和管理與資源相關的成本，並在 AWS Identity and Access Management (IAM) 政策中提供更好的安全性。標記步驟函數資源允許它們由管理AWS Resource Groups。如需有關資源群組的詳細資訊，請參閱AWS Resource Groups使用者指南。

對於以標籤為基礎的授權，如下列範例所示的狀態機器執行資源會繼承與狀態機器相關聯的標籤。

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

當您呼叫DescribeExecution或其他指定執行資源 ARN 的 API 時，Step Functions 會在執行以標籤為基礎的授權時，使用與狀態機器相關聯的標籤來接受或拒絕要求。這可協助您在狀態機器層級允許或拒絕存取狀態機器執行。

若要查看資源標記的相關限制，請參閱 與標記相關的限制。

主題

- 進行標記以分配成本
- 針對安全進行標記

- 在步驟函數主控台中檢視及管理標籤
- 使用步驟函數 API 動作管理標籤

進行標記以分配成本

若要組織和識別用於成本分配的 Step Functions 資源，您可以新增中繼資料標籤，以識別狀態機器或活動的目的。這一點在擁有許多資源時特別實用。您可以使用成本分配標籤來整理您的 AWS 帳單，以反映您自己的成本結構。若要這麼做，請註冊取得 AWS 帳戶帳單，以納入標籤索引鍵和鍵值。如需詳細資訊，請參閱 AWS Billing 使用者指南中的設定每月成本分配報告。

例如，您可以新增代表 Step Functions 資源之成本中心和用途的標籤，如下所示。

資源	索引鍵	值
StateMachine1	Cost Center	34567
	Application	Image processing
StateMachine2	Cost Center	34567
	Application	Rekognition processing
Activity1	Cost Center	12345
	Application	Legacy database

這種標記機制，可讓您將同一個成本中心內執行相關任務的兩個狀態機器，歸為同一組，並且使用不同的成本分配標籤，來標記不相關的活動。

針對安全進行標記

IAM 支援根據標籤控制資源的存取。若要根據標籤控制存取，請在 IAM 政策的條件元素中提供有關資源標籤的資訊。

例如，您可以限制所有 Step Functions 資源的存取權，這些資源包含含索引鍵`environment`和值的標籤`production`。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:DeleteActivity",
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

如需詳細資訊，請參閱《IAM 使用者指南》中的「使用標籤控制存取」。

在步驟函數主控台中檢視及管理標籤

Step Functions 可讓您在 Step Functions 主控台中檢視和管理狀態機器的標籤。從狀態機器的 Details (詳細資訊) 頁面，選擇 Tags (標籤)。您可以在此檢視與您狀態機器相關聯的現有標籤。

> **Note**
>
> 如需管理活動的標籤，請參閱 使用步驟函數 API 動作管理標籤。

若要新增或刪除與您的狀態機器相關的標籤，請選取 Manage Tags (管理標籤) 按鈕。

1. 瀏覽至狀態機器的詳細資訊頁面。
2. 選擇 Tags (標籤)，在 Executions (執行) 和 Definition (定義) 旁邊。
3. 選擇 Manage tags (管理標籤)。
   - 若要修改現存標籤，請編輯Key (索引鍵) 和 Value (值)。
   - 如要移除現有標籤，請選擇 Remove tag (移除標籤)。
   - 如要新增標籤，請選擇 Add tag (新增標籤)，然後輸入 Key (鍵) 和 Value (值)。
4. 選擇 儲存 。

使用步驟函數 API 動作管理標籤

若要使用步驟函數 API 管理標籤，請使用下列 API 動作：

- ListTagsForResource
- TagResource
- UntagResource

AWS Step Functions 流程工作室

Workflow Studio for AWS Step Functions 是低程式碼的視覺化工作流程設計工具，可讓您透過協調服務來建立無伺服器工作流 AWS 程。使用其 drag-and-drop 功能或內建程式碼編輯器，您可以建立和編輯工作流程、控制每個狀態的輸入和輸出篩選或轉換方式，以及設定錯誤處理。當您拖放狀態以建立工作流程時，Workflow Studio 會驗證您的工作並自動產生程式碼。您可以在程式碼編輯器中檢閱產生的程式碼或更新狀態機定義。完成後，您可以儲存工作流程、執行工作流程，然後在 Step Functions 主控台中檢查結果。您可以視覺化地新增和修改工作流程，以協調應用程式中的多個服務。

若要使用 Step Functions 工作流程 Studio，您將需要和認證 AWS 帳戶，以提供您想要使用的任何資源的正確權限。如需詳細資訊，請參閱 開始使用的先決條件 AWS Step Functions。

> **Note**
>
> 工作流工作室不支援互聯網資源管理器 11. 如果您使用互聯網資源管理器 11 和使用工作流程工作室遇到問題，請嘗試使用不同的瀏覽器。

當您在 Step Functions 中建立或編輯工作流程時，您可以從步 Step Functions 主控台存取工作流程 Studio。您也可以在中存取工作流程工作室AWS 應用程式編寫器。工作流 Studio 中Application Composer提供了一個可視化的 IaC 環境，使您可以輕鬆地將工作流程納入使用 IaC 工具，如模板構建的無服務器應用程序。AWS CloudFormation使用中的工作流程 StudioApplication Composer，您可以使用AWS CloudFormation範本來建置工作流程。在中Application Composer，您可以新增工作流程、修改現有工作流程，以及將個別工作流程步驟連接至其他應用程式資源。 Application Composer自動創建和更新所需的CloudFormation資源和配置。這可協助您在單一位置建立和管理工作流程中使用的所有資源。這也可協助您加速從工作流程原型設計到生產部署的過程。

當您在中使用工作流程 Studio 時Application Composer，您也可以直接連接到您的本端專案。這有助於您與視覺畫布一起在整合式開發環境 (IDE) 中工作。如需詳細資訊，請參閱 使用工作流程工作室 Application Composer。

主題

- 介面概觀
- 使用工作流程
- 為您的狀態配置輸入和輸出
- 工作流程中的執行角色
- 錯誤處理

- 自學課程：學習使用 AWS Step Functions 工作流程工作室

介面概觀

Workflow Studio for AWS Step Functions 是低程式碼的視覺化工作流程設計工具，可讓您透過協調建立無伺服器工作流程。 AWS 服務借助其拖放功能，Workflow Studio 使您可以輕鬆構建，編輯和可視化工作流程原型。工作流程 Studio 還提供了一個內置的代碼編輯器，用於編寫和編輯使用Amazon States Language（ASL）Step Functions 控制台中的工作流定義。

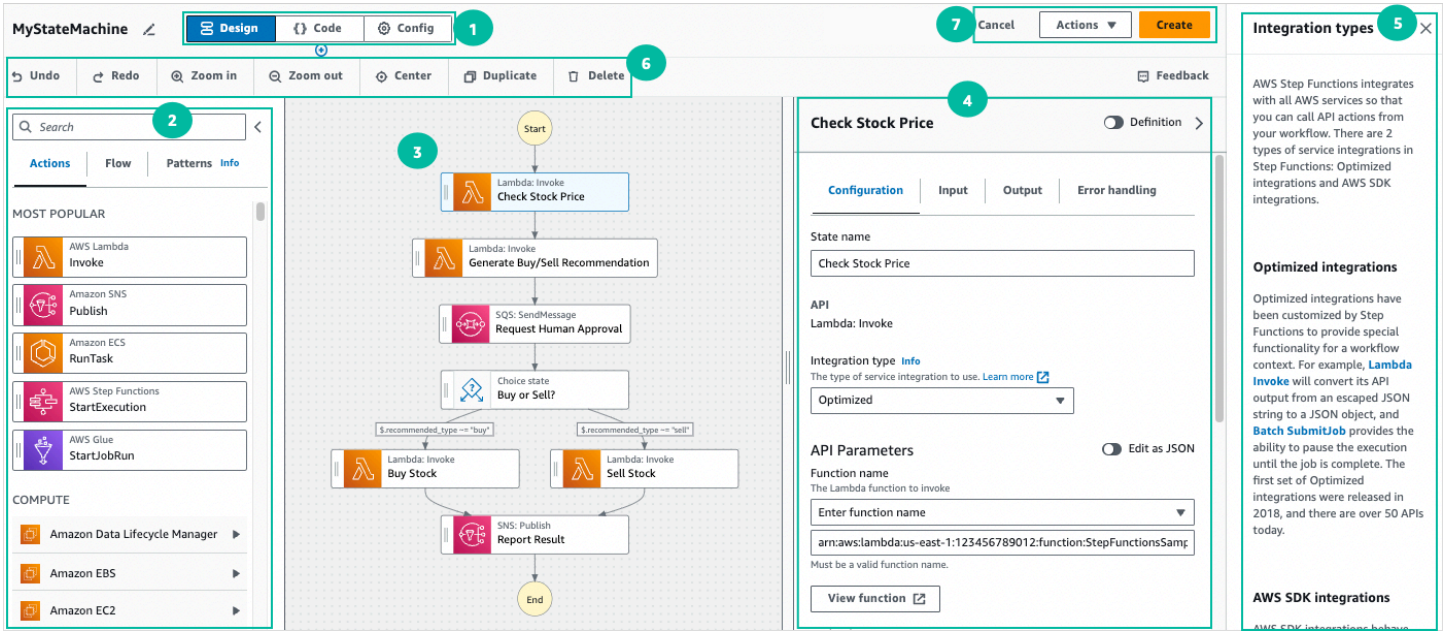
為了幫助您構建和視覺化您的工作流程，編輯它們的定義，並管理它們的 Config，Workflow Studio 提供了三種模式：設計，代碼和配置。以下各節將詳細說明這些模式。

在本主題中

- 設計模式
- 程式碼模式
- Config 模式
- 鍵盤快速鍵

設計模式

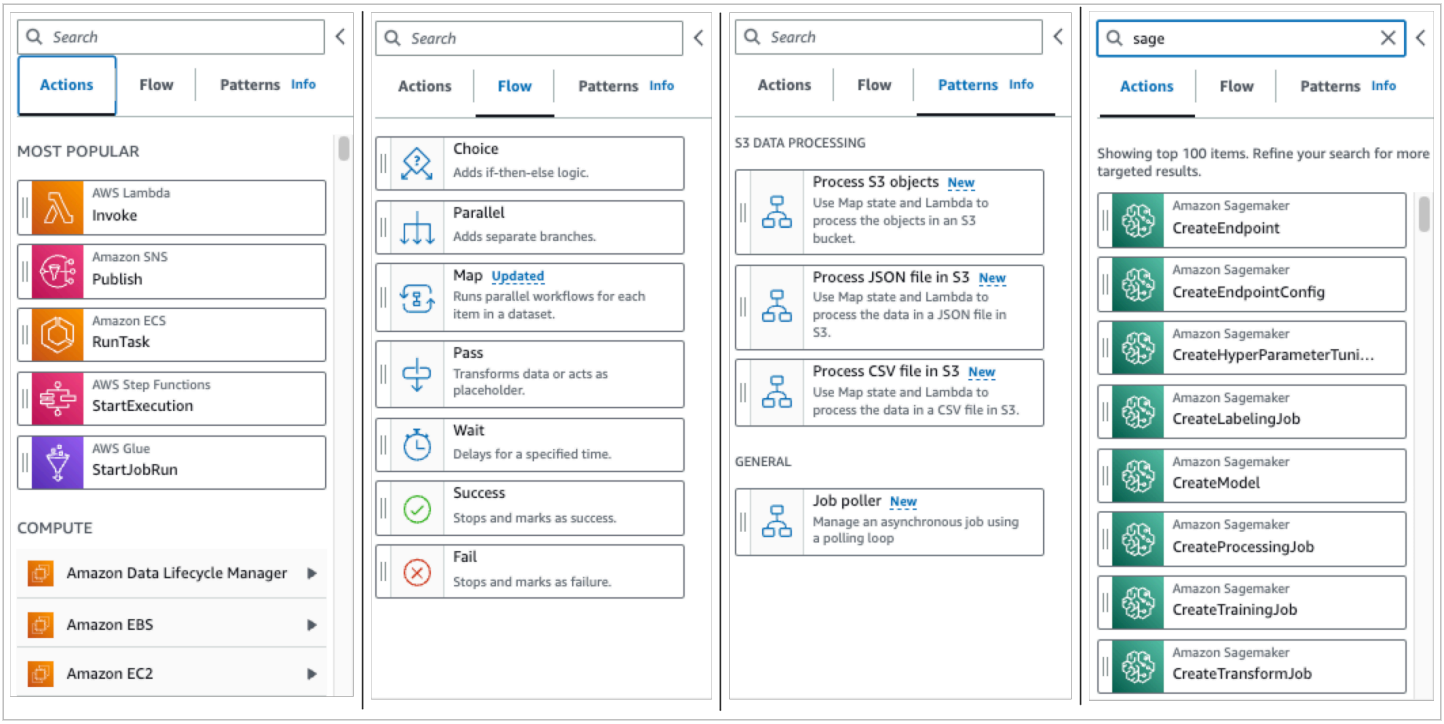
工作流程 Studio 的設計模式提供了一個圖形化界面，以便在您構建其原型時視覺化您的工作流程。以下影像展示了「設計」模式中可用的不同元件。



1. 模式按鈕 — 使用模式按鈕在工作流 Studio 的 [設計]、[程式碼] 和 [設 Config] 模式之間切換。如果工作流程的 ASL 定義中的 JSON 無效，則無法切換模式。
2. 狀態瀏覽器包含下列三個索引標籤：
   - [動作] 索引標籤提供 AWS API 清單，您可以將這些 API 拖放到畫布中的工作流程圖形中。每個動作代表一個任務狀態。
   - 「流程」索引標籤提供流程狀態清單，您可以將其拖放到畫布中的工作流程圖表中。
   - [模式] 索引標籤提供數個可重複使用的建置區塊 ready-to-use，可用於各種使用案例。例如，您可以使用這些模式在 Amazon S3 儲存貯體中反覆處理資料。
3. 您可以將狀態拖放到工作流程圖表中、變更狀態順序，以及選取要設定或檢視的狀態。Canvas
4. 您可以在此Inspector面板中檢視和編輯您在畫布上選取的任何狀態的屬性。開啟定義切換以檢視工作流程的 Amazon 州語言代碼，並反白顯示目前選取的狀態。
5. 當您需要協助時，資訊連結會開啟包含內容資訊的面板。這些面板也包括 Step Functions 文件中相關主題的連結。
6. 設計工具列 — 包含一組用於執行常用動作的按鈕，例如復原、刪除和放大。
7. 公用程式按鈕 — 一組用來執行工作的按鈕，例如儲存工作流程或將其 ASL 定義匯出至 JSON 或 YAML 檔案。

狀態瀏覽器

您可以在「狀態」瀏覽器中選取要拖放到工作流程圖表中的狀態。「動作」索引標籤提供 AWS API 清單，而「流程」索引標籤則提供流程狀態清單。雖然「模式」索引標籤提供了數個可重複使用的建置區塊 ready-to-use，可用於各種使用案例。您可以使用頂端的「搜尋」方塊，在「狀態瀏覽器」中搜尋所有狀態。



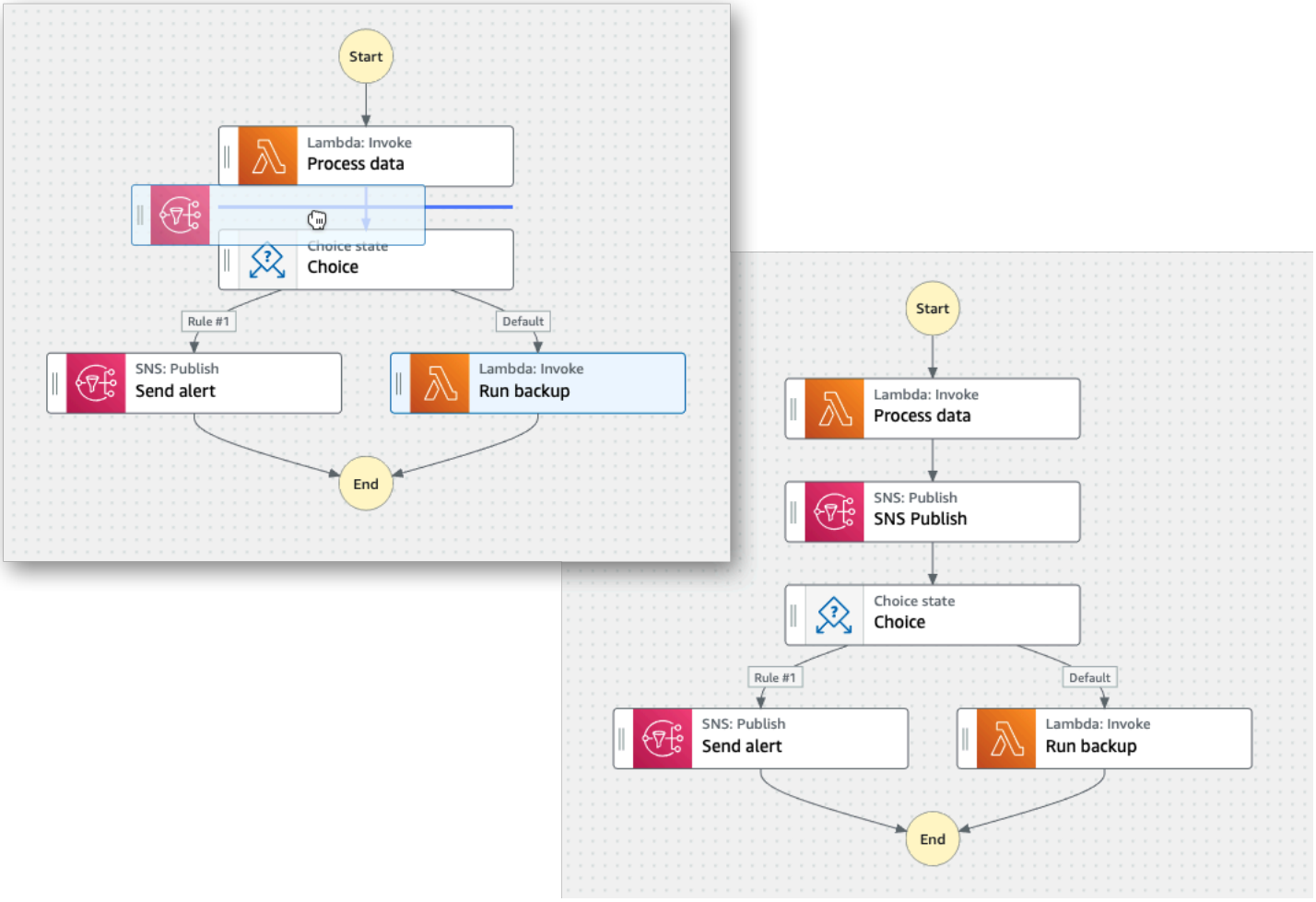
您可以使用七種流程狀態來引導和控制工作流程。所有這些都取自先前狀態的輸入，許多可讓您篩選前一個狀態的輸入，並將輸出篩選為後續狀態。流量狀態為：

- Choice：將執行分支之間的選擇添加到您的工作流程。在「Inspector」的「組態」索引標籤中，您可以設定規則以決定工作流程將轉換至哪個狀態。
- 平行：將執行的 parallel 分支新增至您的工作流程。
- Map：動態迭代輸入數組的每個元素的步驟。與`Parallel`流動狀態不同，`Map`狀態會針對狀態輸入中陣列的多個項目執行相同的步驟。
- Pass：可讓您將其輸入傳遞至輸出。（可選）您可以將固定數據添加到輸出中。
- 等候：讓您的工作流程暫停一段時間，或直到指定的時間或日期。
- Succeed：成功停止您的工作流程。
- Fail：停止失敗的工作流程。

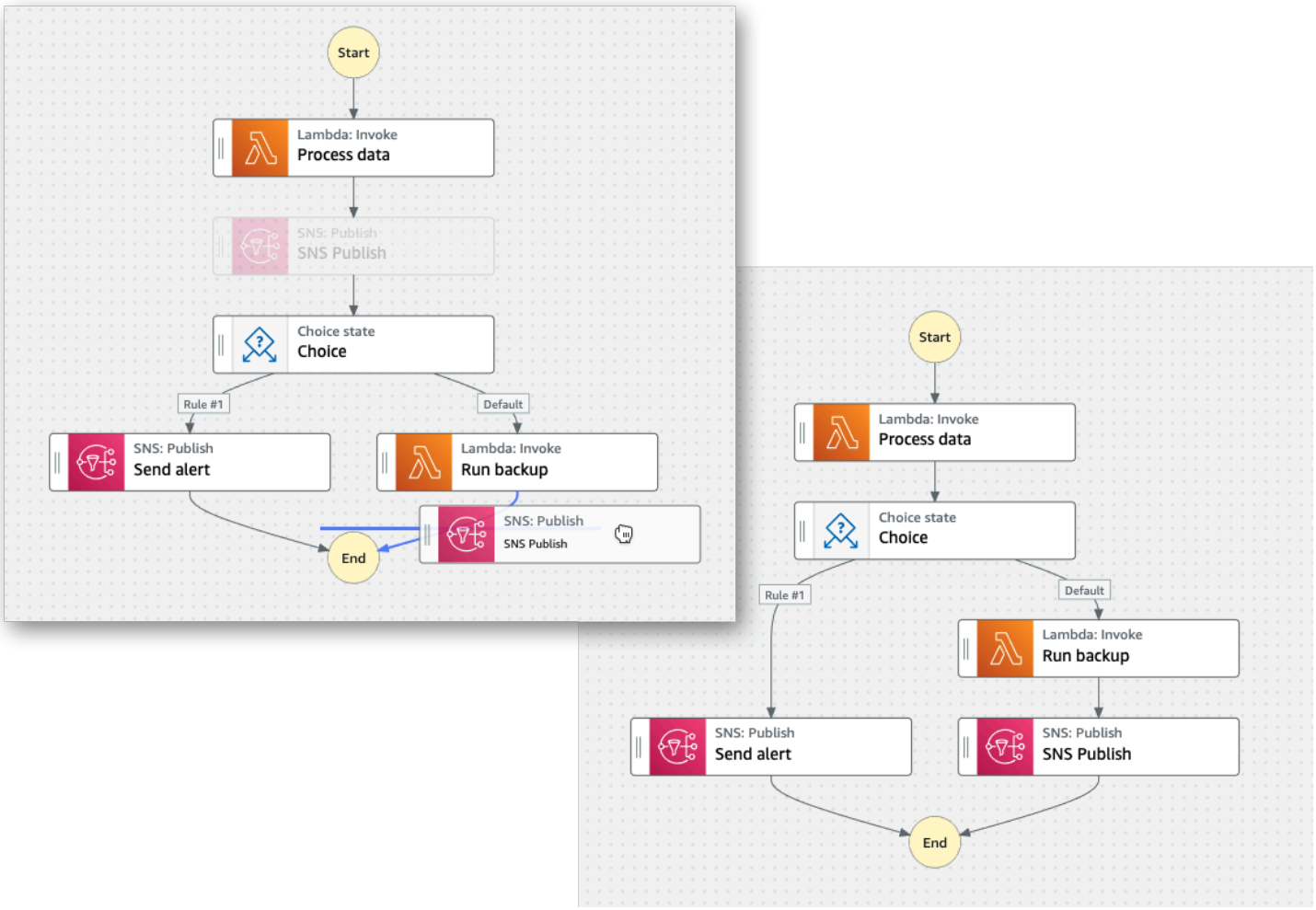
Canvas

選擇要新增至工作流程的狀態後，請將其拖曳至畫布，然後放入工作流程圖表中。您也可以拖放狀態，將狀態移至工作流程中的不同位置。如果您的工作流程很複雜，您可能無法在畫布面板中檢視所有工作流程。使用畫布頂端的控制項來放大或縮小。若要檢視工作流程圖形的不同部分，您可以在畫布中拖曳工作流程圖表。

將工作流程狀態從「動作」或「流程」標籤拖放到工作流程中。有一條線會顯示它將放置在工作流程中的位置。新的工作流程狀態已新增至您的工作流程，其程式碼會自動產生。

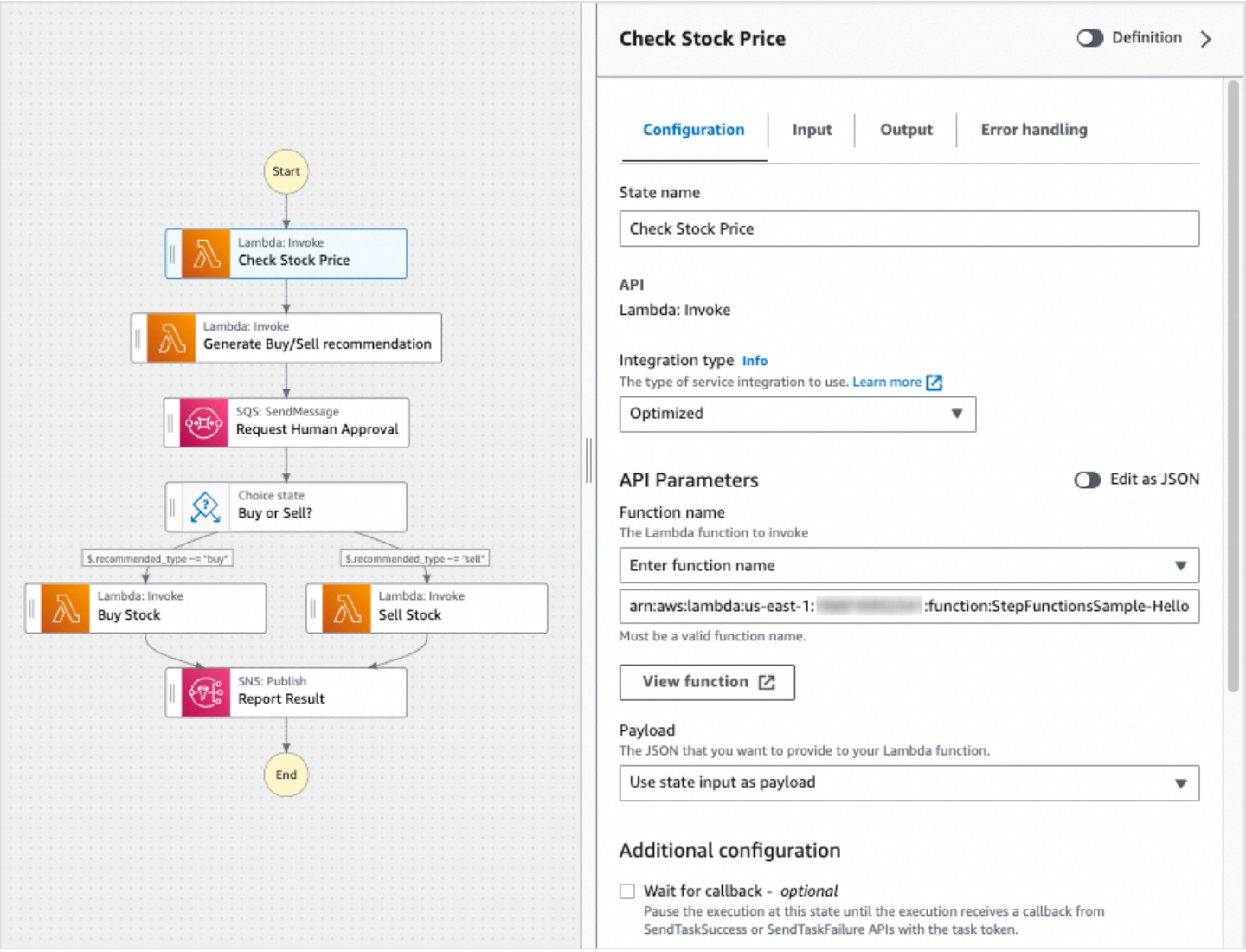


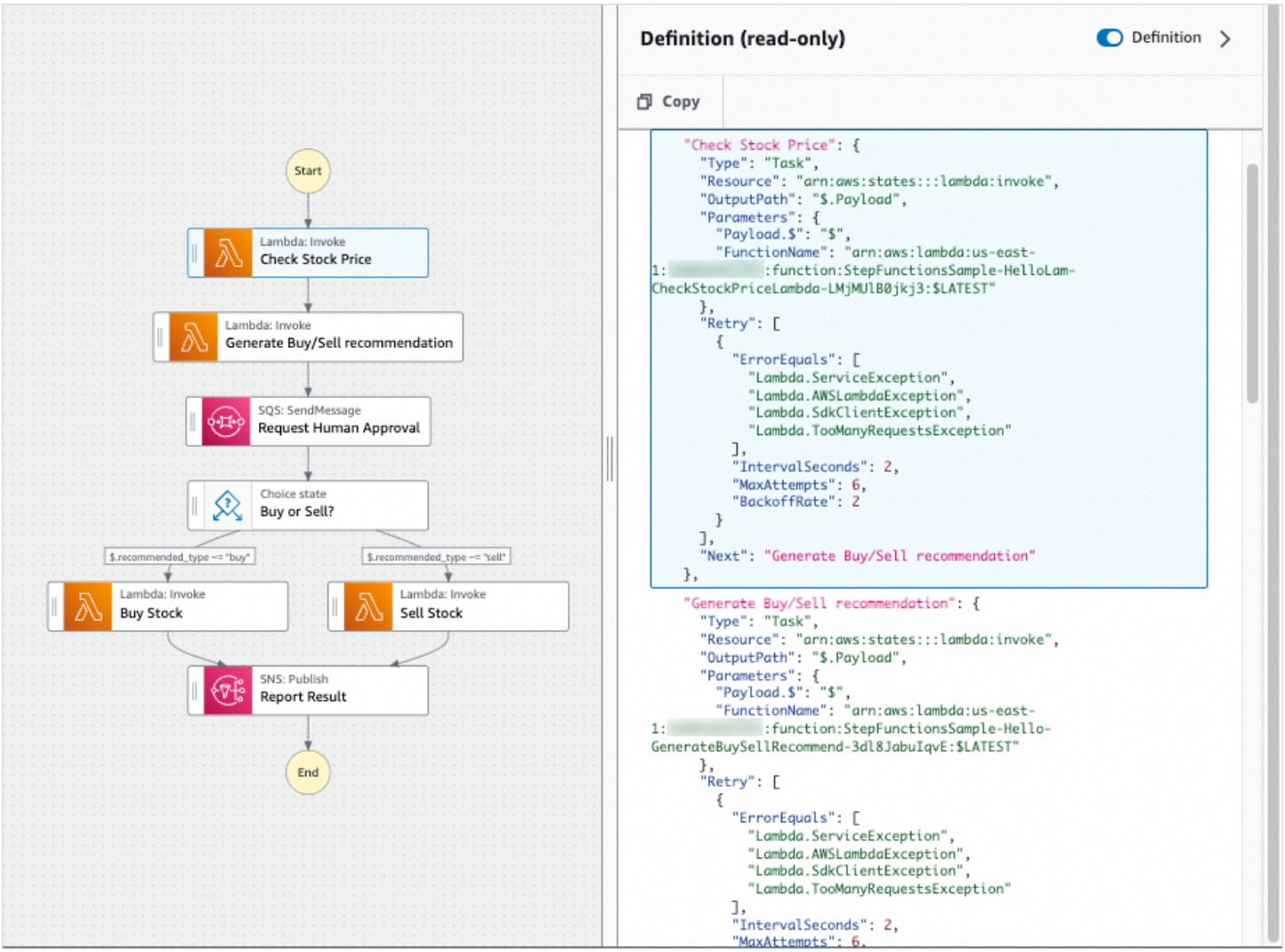
若要變更狀態的順序，您可以將其拖曳至工作流程中的其他位置。



Inspector

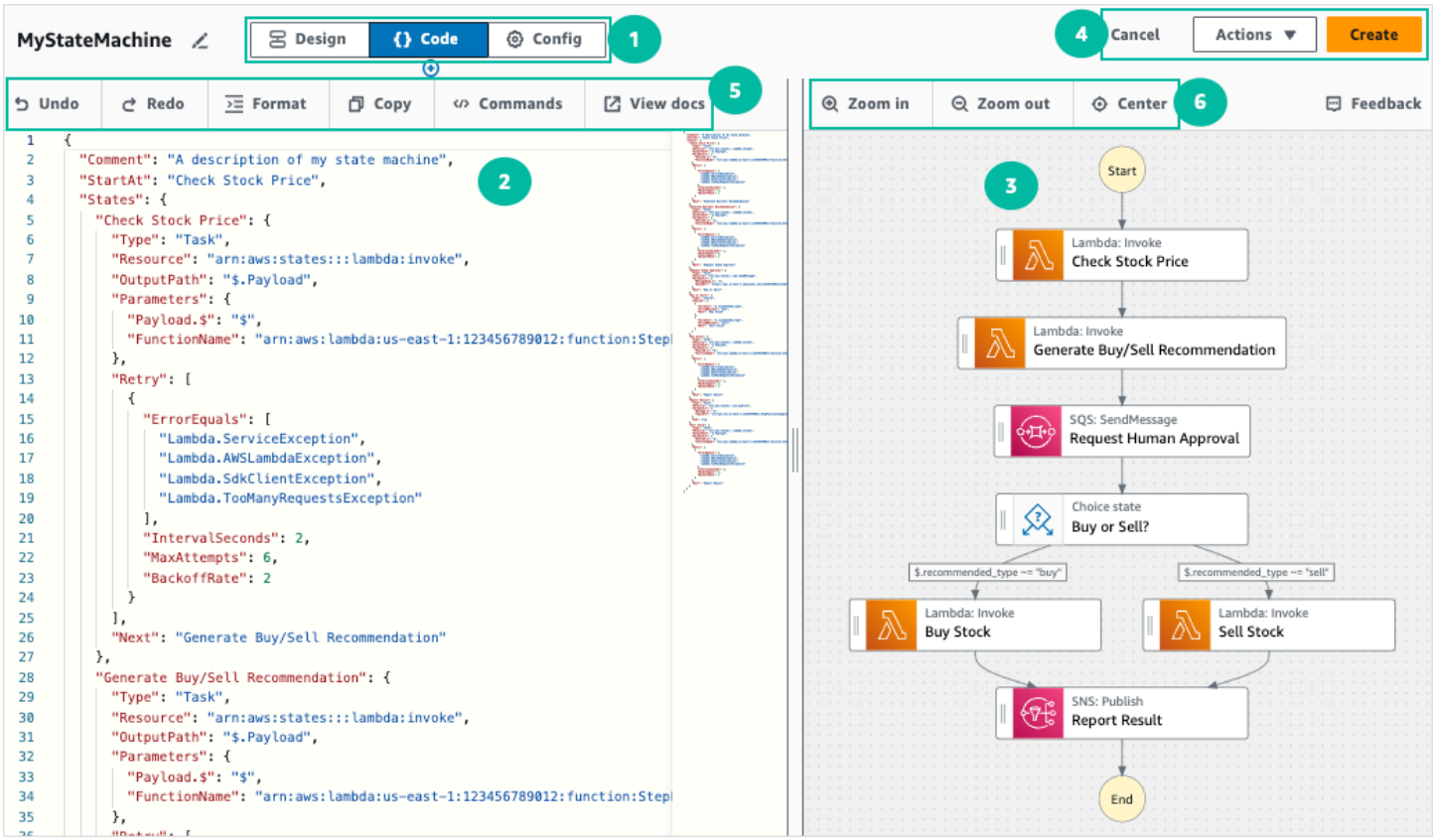
您可以設定新增至工作流程的任何狀態。選擇您要配置的狀態，您將在 Inspector 面板中看到其配置選項。若要查看工作流程程式碼自動產生的 ASL 定義，請開啟定義切換。與您選取的狀態相關聯的 ASL 定義將會反白顯示。


Start
Lambda: Invoke
Check Stock Price
Lambda: Invoke
Generate Buy/Sell recommendation
SQS: SendMessage
Request Human Approval
Choice state
Buy or Sell?
$.recommended_type == "buy"
$.recommended_type == "sell"
Lambda: Invoke
Buy Stock
Lambda: Invoke
Sell Stock
SNS: Publish
Report Result
End
Check Stock Price
Definition
Configuration
Input
Output
Error handling
State name
Check Stock Price
API
Lambda: Invoke
Integration type Info
The type of service integration to use. Learn more
Optimized
API Parameters
Edit as JSON
Function name
The Lambda function to invoke
Enter function name
arn:aws:lambda:us-east-1:
:function:StepFunctionsSample-Hello
Must be a valid function name.
View function
Payload
The JSON that you want to provide to your Lambda function.
Use state input as payload
Additional configuration
Wait for callback - optional
Pause the execution at this state until the execution receives a callback from SendTaskSuccess or SendTaskFailure APIs with the task token.



程式碼模式

Workflow Studio 的程式碼模式提供整合的程式碼編輯器，可在 Step Functions 主控台中檢視、撰寫及編輯工作流程的 Amazon States Language (ASL) 定義。下列影像顯示「程式碼」模式中可用的不同元件。



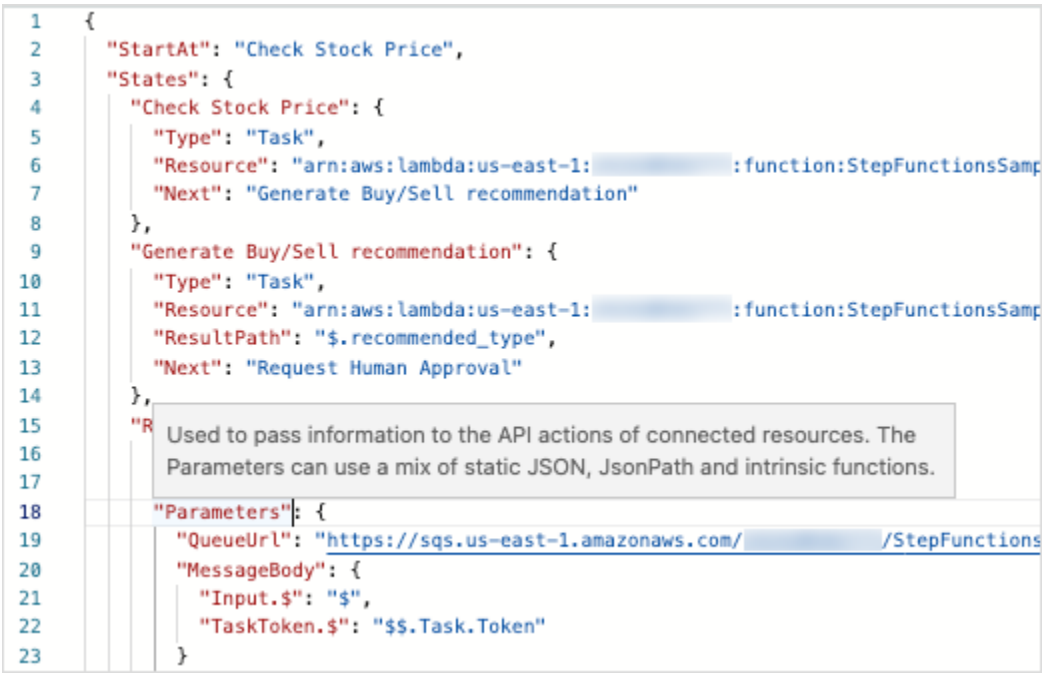
1. 模式按鈕 — 使用模式按鈕在工作流 Studio 的 [設計]、[程式碼] 和 [設 Config] 模式之間切換。如果工作流程的 ASL 定義中的 JSON 無效，則無法切換模式。
2. 您可以程式碼編輯器在工作流程 Studio 中撰寫和編輯工作流程的 ASL 定義。程式碼編輯器也提供了一些功能，例如語法醒目提示和自動完成功能。
3. 圖形視覺化窗格— 顯示工作流程的實時圖形可視化。
4. 公用程式按鈕 — 一組用來執行工作的按鈕，例如儲存工作流程或將其 ASL 定義匯出至 JSON 或 YAML 檔案。
5. 程式碼工具列 — 包含一組用來執行常用動作的按鈕，例如復原動作或格式化程式碼。
6. 圖形工具列 — 包含一組用於執行常用動作的按鈕，例如放大和縮小工作流程圖形。

程式碼編輯器

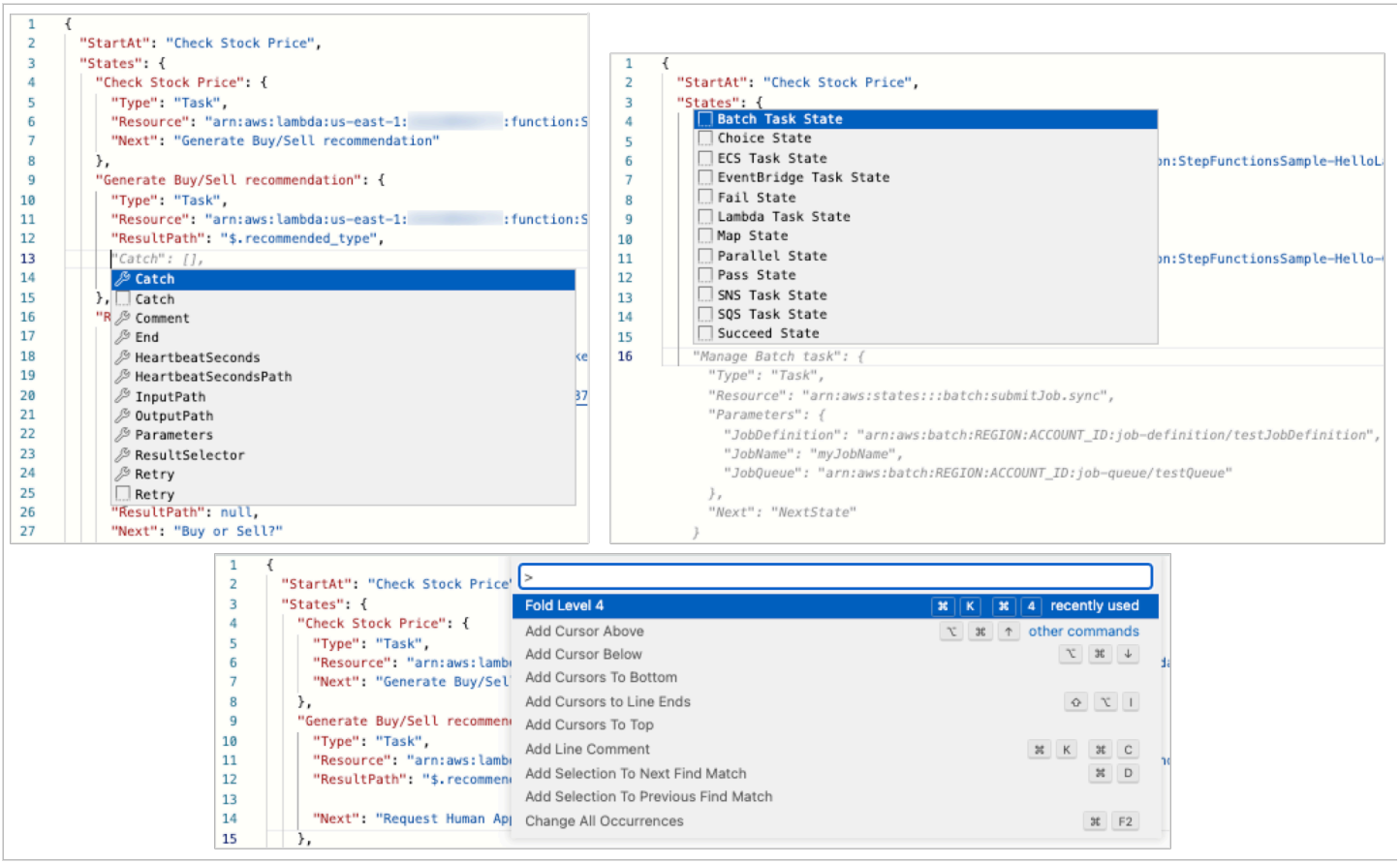
程式碼編輯器提供類似 IDE 的體驗，可在工作流程 Studio 中使用 JSON 撰寫和編輯工作流程定義。程式碼編輯器包含數個功能，例如語法醒目提示、自動完成建議、ASL 定義驗證，以及內容感應式說明顯示。當您更新工作流程定義時，會圖形視覺化窗格呈現工作流程的即時圖表。您也可以在中看到更新的工作流程圖形設計模式。

如果您在設計模式或圖形視覺化窗格中選取狀態，該狀態的 ASL 定義會在程式碼編輯器中反白顯示。如果您在「設計」模式或圖形視覺化窗格中重新排序、刪除或新增狀態，工作流程的 ASL 定義會自動更新。

將游標置於工作流程定義中的任何欄位上，以檢視其上下文相關說明作為工具提示。



自動完成建議會顯示您可以包含在工作流程中的欄位或狀態的程式碼片段。若要查看可包含在特定狀態中的欄位清單，請按**Ctrl+Space**。若要為工作流程中的新狀態產生程式碼片段，請在目前狀態的定義**Ctrl+Space**之後按下。您也可以按下**F1**以顯示可用指令的清單。



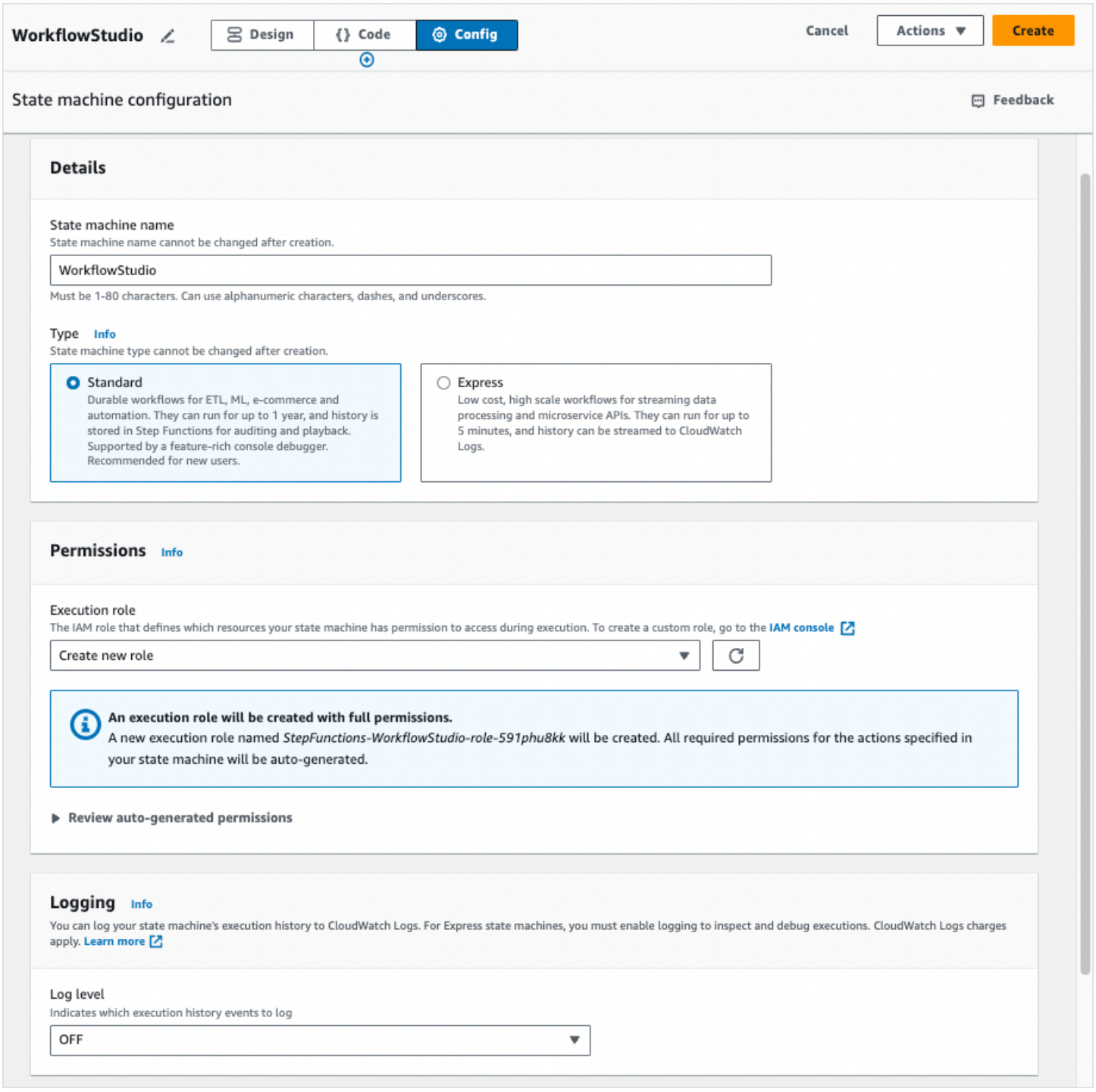
圖形視覺化窗格

圖形視覺效果可讓您以圖形格式查看工作流程的外觀。當您在 Workflow Studio 中撰寫工作流程定義時，圖形視覺化窗格會呈現工作流程的即時圖形。程式碼編輯器當您在圖形視覺化窗格中重新排序、刪除或複製狀態時，程式碼編輯器中的工作流程定義會自動更新。同樣地，當您在程式碼編輯器中更新工作流程定義、重新排序、刪除或新增狀態時，視覺效果也會自動更新。

如果工作流程 ASL 定義中的 JSON 無效，圖形視覺化窗格會暫停轉譯，並在窗格底部顯示狀態訊息。

Config 模式

工作流程 Studio 的 Config 模式可讓您管理狀態機器的配置。在此模式中，您可以為狀態機器指定詳細資料，例如狀態機器名稱及其類型、IAM 許可和狀態機器的記錄組態。您可以在此模式中指定的其他其他組態包括在建立狀態機器時啟用 AWS X-Ray 追蹤和發佈版本。建立狀態機之後，您可以編輯狀態機器名稱和類型以外的所有狀態機組態選項。下圖顯示了您可以在「組態」模式中指定的一些組 Config。



管理狀態機組態

若要管理您的狀態機組態，請執行下列動作：

1. 在狀態機器名稱方塊中輸入狀態機的名稱。

> **Tip**
>
> 或者，選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 下，指定名稱。

> **Important**
>
> 建立狀態機器之後，您無法編輯狀態機器名稱。

2. 在類型中，選擇標準或快速的狀態機器類型。如需狀態機器類型的相關資訊，請參閱標準與快速工作流程。

> **Important**
>
> 建立狀態機器之後，您無法編輯狀態機器類型。

3. 在許可中，選取要用作狀態機器執行角色的 IAM 角色。

   - 建立新角色 (建議使用)：如果選取此選項，Step Functions 會以建立狀態機器時所需的最低權限，自動為狀態機器建立執行角色。這些自動產生的 IAM 角色對於您 AWS 區域 在其中建立狀態機器有效。

     > **Tip**
     >
     > 若要檢閱 Step Functions 將為狀態機器自動產生的權限，請選擇 [檢閱自動產生的權限]。

     > **Note**
     >
     > 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

   - 選擇現有角色：為狀態機器建立您自己的 IAM 角色，然後從下面列出的選項中選擇「選擇現有角色」。確保角色的策略包含您希望狀態機器承擔的權限。

     如需有關建立和編輯 IAM 政策的資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策。

- 輸入角色 ARN：指定要用於此狀態機器的現有 IAM 角色的 Amazon 資源名稱 (ARN)。例如 `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`。

4. 在記錄中，設定狀態機器的記錄層級。Step Functions 會根據您的選擇記錄執行歷史記錄事件。您可以選擇以下其中一個選項：
   - ALL：會記錄所有事件類型。
   - 錯誤：會記錄所有錯誤事件類型，例如 TaskFailed 和 ExecutionFailed。
   - 致命：記錄所有嚴重錯誤事件類型，例如 ExecutionAborted 和 ExecutionFailed。
   - OFF：不會記錄任何事件類型。

   如需記錄層級的詳細資訊，請參閱日誌層級。
5. 在其他組態中，設定下列一或多個選用組態：
   - 啟用X-Ray追蹤：針對狀態機器執行傳送追蹤的 Step Functions 選擇此核取方塊，即使追蹤 ID 未由上游服務傳送追蹤。X-Ray如需詳細資訊，請參閱 AWS X-Ray 和 Step Functions。
   - 建立時發佈版本：版本是您可以執行的狀態機器的編號、不可變快照集。選擇此核取方塊可在建立狀態機時發佈狀態機的版本。Step Functions 發布版本 1 作為狀態機的第一個修訂版。

     如需有關版本的詳細資訊，請參閱狀態機版本。
   - 新增標籤：選擇此方塊可將標籤新增至狀態機。新增標籤可協助您追蹤和管理與資源相關的成本，並在 IAM 政策中提供更好的安全性。如需標籤的詳細資訊，請參閱步驟函數中的標記。
6. 選擇建立。
7. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色組態] 回到 [設定] 模式。

鍵盤快速鍵

工作流程工作室支援下列鍵盤快速鍵：

鍵盤快速鍵	函式
Shortcuts for the Code mode	
Ctrl+space	Auto-complete suggestions

鍵盤快速鍵	函式
F1	Display a list of available commands
Common shortcuts for the Design and Code modes	
Ctrl+Z	Undo the last operation
Ctrl+Shift+Z	Redo the last operation
Alt+C	Center the workflow in the canvas
Backspace	Remove all selected states
Delete	Remove all selected states
Ctrl+D	Duplicate selected state

使用工作流程

了解如何使用 Step Functions 工作流程 Studio 建立、編輯和執行工作流程。工作流程準備就緒後，您可以將其匯出。您也可以使用工作流程 Studio 進行快速原型製作。

在本主題中

建立工作流程

在 Workflow Studio 中，您可以選擇初學者範本，或選擇空白範本以從頭開始建立工作流程。對於空白範本，您可以使用「設計」或「程式碼」模式來建立工作流程。

入門範本是一個 ready-to-run 範例專案，可自動建立工作流程 proptotype 和定義，並將專案所需的所有相關 AWS 資源部署到. AWS 帳戶您可以使用這些初學者範本依原樣部署和執行它們，或使用工作流程原型來建置它們。如需有關初學者範本的更多資訊，請參閱Step Functions 的範例專案。

使用初學者範本建立工作流程

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在 [選擇範本] 對話方塊中，執行下列其中一項作業以選擇範例專案，例如 [工作計時器] 範例專案：

   - **`Task Timer`**在 [依關鍵字搜尋] 方塊中輸入，然後從傳回的搜尋結果中選擇 [工作計時器]。
   - 瀏覽右窗格中「全部」下列出的範例專案，然後選擇「工作計時器」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。
5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。在中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或切換至以程式碼模式更新工作流程的 Amazon States Language (ASL) 定義。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

     > **Tip**
     >
     > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

> **Note**
>
> 建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

> **Important**
>
> CloudFormation 範本中使用的每項服務均需支付標準費用。

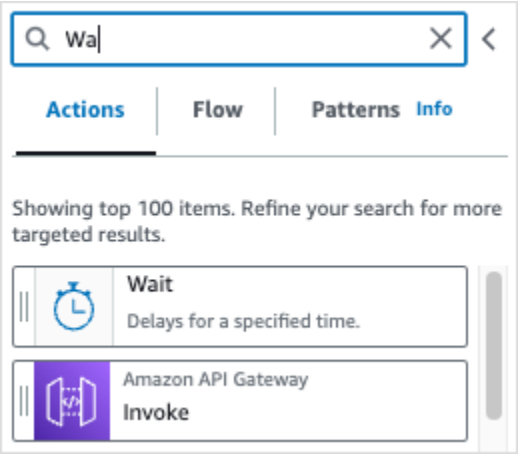
使用空白範本建立工作流程

1. 開啟「Step Functions」主控台。
2. 選擇 Create state machine (建立狀態機器)。
3. 在「選擇範本」對話方塊中，選取「空白」。
4. 選擇選取。這會在設計模式中開啟工作流程工作室

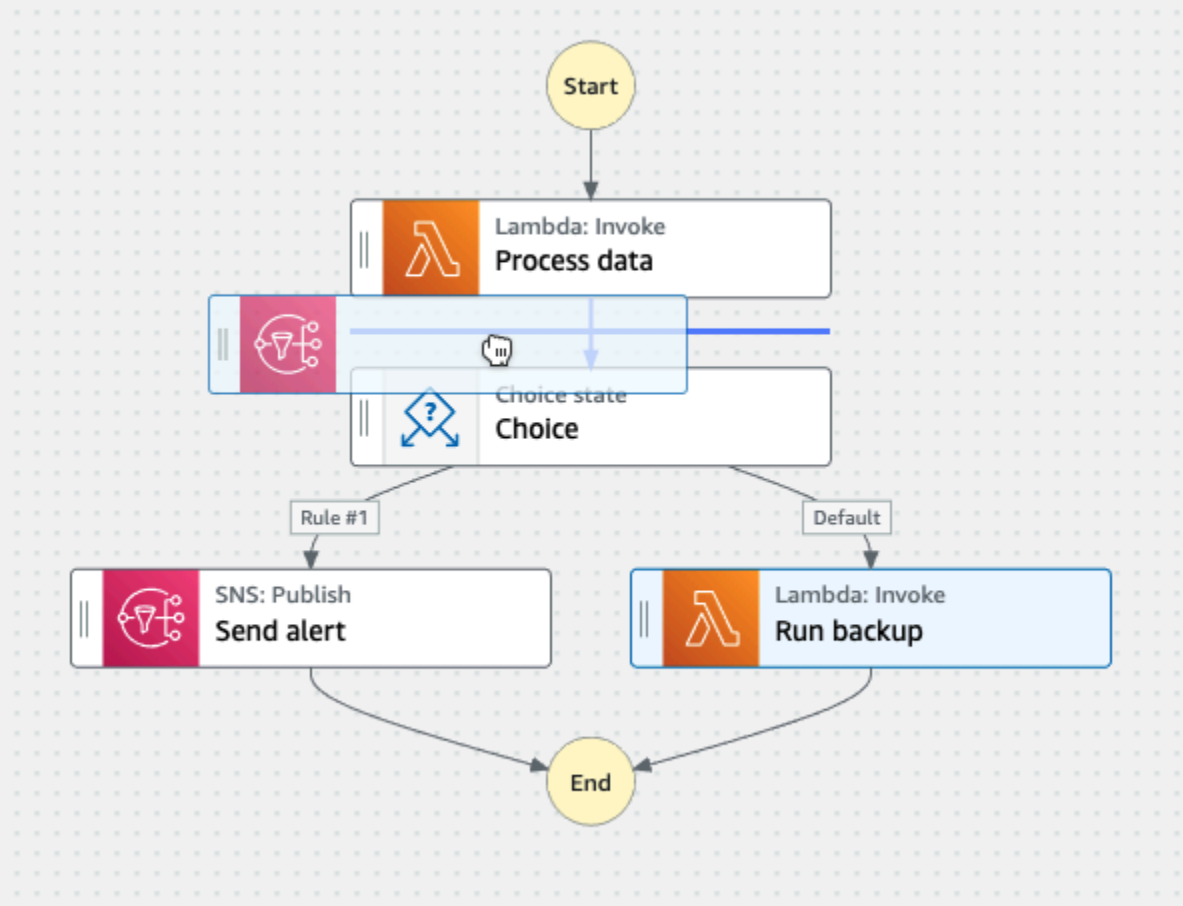
   您現在可以開始在中設計工作流程，設計模式或在中編寫工作流程定義程式碼模式。
5. 選擇「Config 定」以管理中工作流程的組態Config 模式。例如，提供工作流程的名稱並選擇其類型。

設計工作流程

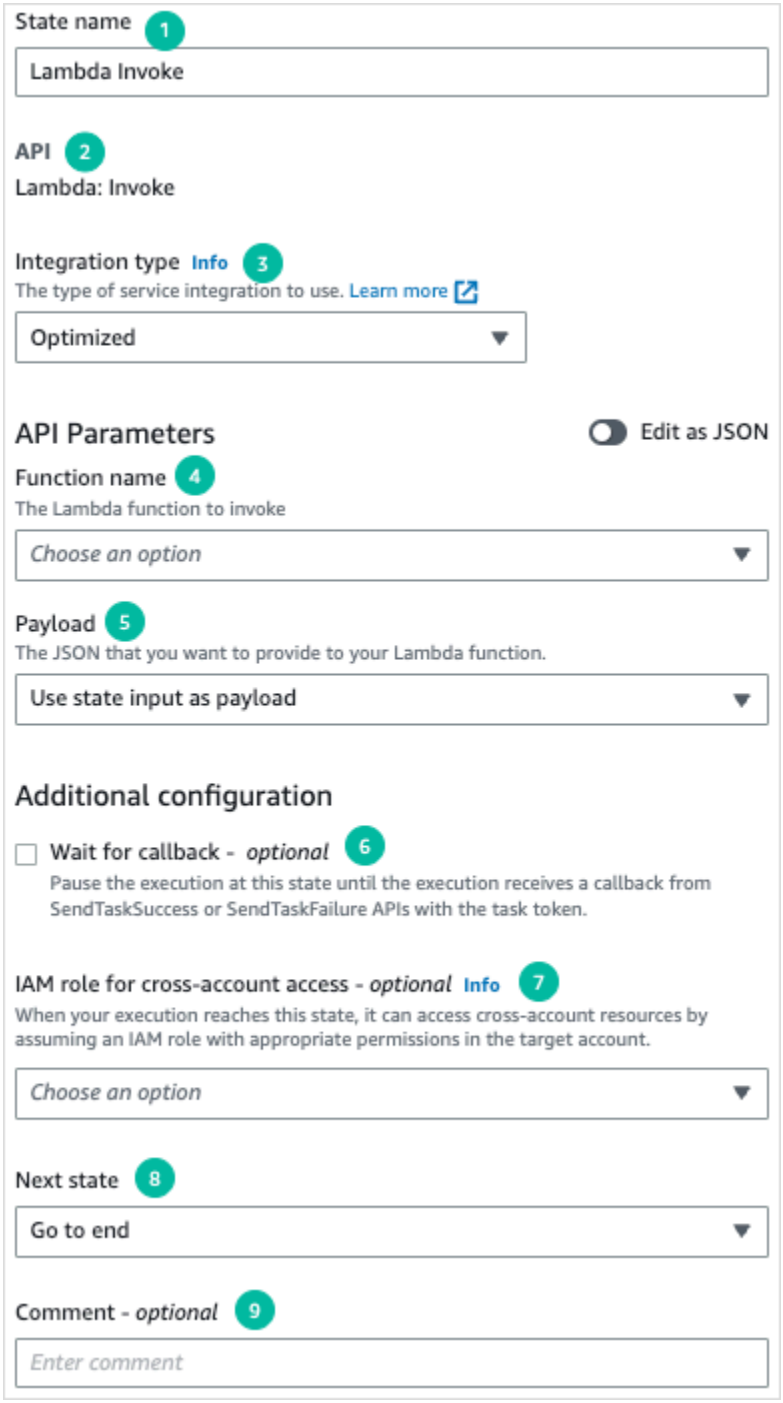
如果您知道要新增的狀態名稱，請使用頂端的搜尋方塊，在的「動作」和「流程」標籤中尋找該狀態設計模式。狀態瀏覽器



否則，請從狀態瀏覽器中選擇狀態，然後將其拖放到畫布上，將其放置在工作流程中所需的位置。您也可以將工作流程中的狀態拖曳至工作流程中的其他位置來重新排序狀態。當您將狀態拖曳到畫布上時，會出現一條線條，您可以將其放置在工作流程中的任何位置。將狀態拖放到畫布上後，其代碼將自動生成並添加到您的工作流程定義中。若要查看定義，請開啟「檢查 Inspector」面板上的「定義」切換開關。若要編輯工作流程定義，請選擇提供整合式程式碼編輯器的項目程式碼模式。



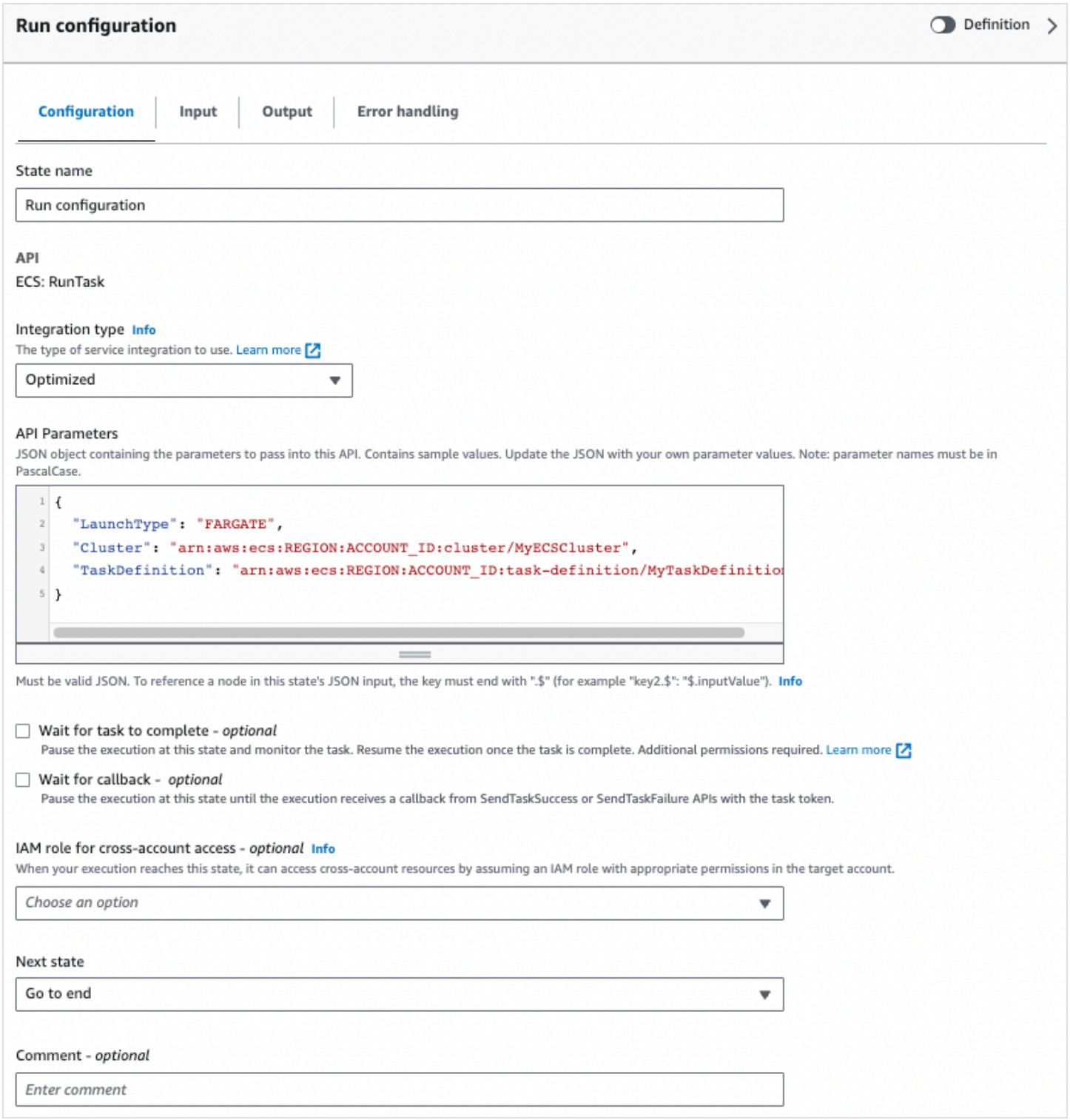
將狀態拖放到畫布上後，您可以在右側的Inspector面板中對其進行配置。此面板包含您放置在畫布上的每個狀態或 API 動作的「組態」、「輸入」、「輸出」和「錯誤處理」索引標籤。您可以在「組態」索引標籤中配置包含在工作流程中的狀態。例如，Lambda 叫用 API 動作的「組態」索引標籤包含下列選項：



1. 狀態名稱可識別狀態。您可以使用自己的名稱或接受預設產生的名稱。
2. 該 API 顯示狀態使用的 API 操作。

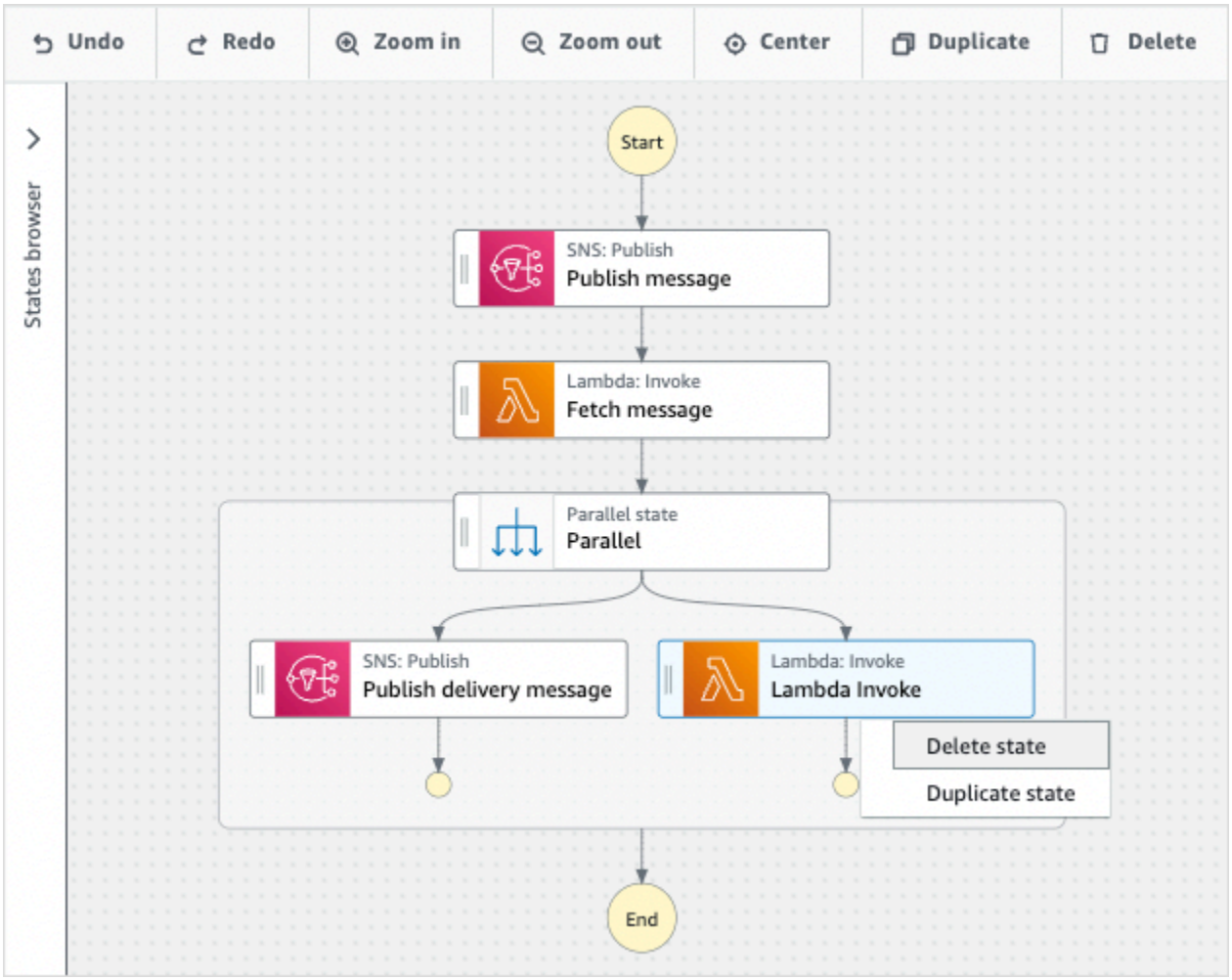
3. 「整合類型」下拉式清單提供選項，讓您選擇 Step Functions 中可用的服務整合類型。您選擇的整合類型可用來呼叫工作流程 AWS 服務 中特定項目的 API 動作。
4. 函數名稱提供的選項可用於：
   - 輸入函數名稱：您可以輸入函數名稱或其 ARN。
   - 在運行時從狀態輸入獲取函數名稱：您可以使用此選項根據指定的路徑從狀態輸入動態獲取函數名稱。
   - 選擇功能名稱：您可以直接從您的帳戶和地區可用的功能中選擇。
5. 「承載」可讓您從下列選項中選取：
   - 使用狀態輸入作為有效負載：您可以使用此選項將狀態的輸入作為提供的有效負載傳遞給 Lambda 函數。
   - 輸入您自己的承載：您可以使用此選項建構 JSON 物件，以作為承載傳遞至 Lambda 函數。此 JSON 可包含靜態值和從狀態輸入中選取的值。
   - 無承載：如果您不想將任何承載傳遞給 Lambda 函數，則可以使用此選項。
6. (選擇性) 某些狀態可以選取 [等待工作完成] 或 [等待回呼]。如果可用，這些選項會選取下列其中一個服務整合模式：
   - 未選取任何選項：Step Functions 將使用請求回應整合模式。Step Functions 將等待 HTTP 響應，然後進展到下一個狀態。Step Functions 不會等待工作完成。當沒有可用的選項時，狀態將使用此模式。
   - 等待任務完成：Step Functions 將使用執行任務 (.sync)集成模式。
   - 等待回調：Step Functions 將使用等候傳回任務字符的回呼集成模式。
7. (選擇性) 若要存取工作流程 AWS 帳戶 中不同設定的資源，Step Functions 提供跨帳戶存取權。跨帳戶存取的 IAM 角色提供下列選項：
   - 提供 IAM 角色 ARN：指定包含適當資源存取權限的 IAM 角色。這些資源可在目標帳戶中使用，目標帳戶是 AWS 帳戶 您進行跨帳戶呼叫的帳戶。
   - 在運行時從狀態輸入獲取 IAM 角色 ARN：在包含 IAM 角色的狀態的 JSON 輸入中指定現有鍵值對的參考路徑。
8. 「下一個狀態」可讓您選取要轉換到下一個狀態的狀態。
9. (選擇性)「註解」欄位可用來新增您自己的註解。它不會影響工作流程，但可用於註釋您的工作流程。

某些狀態會有更多通用的組態選項。例如，Amazon ECS RunTask 狀態組態包含填入預留位置值的`API Parameters`欄位。



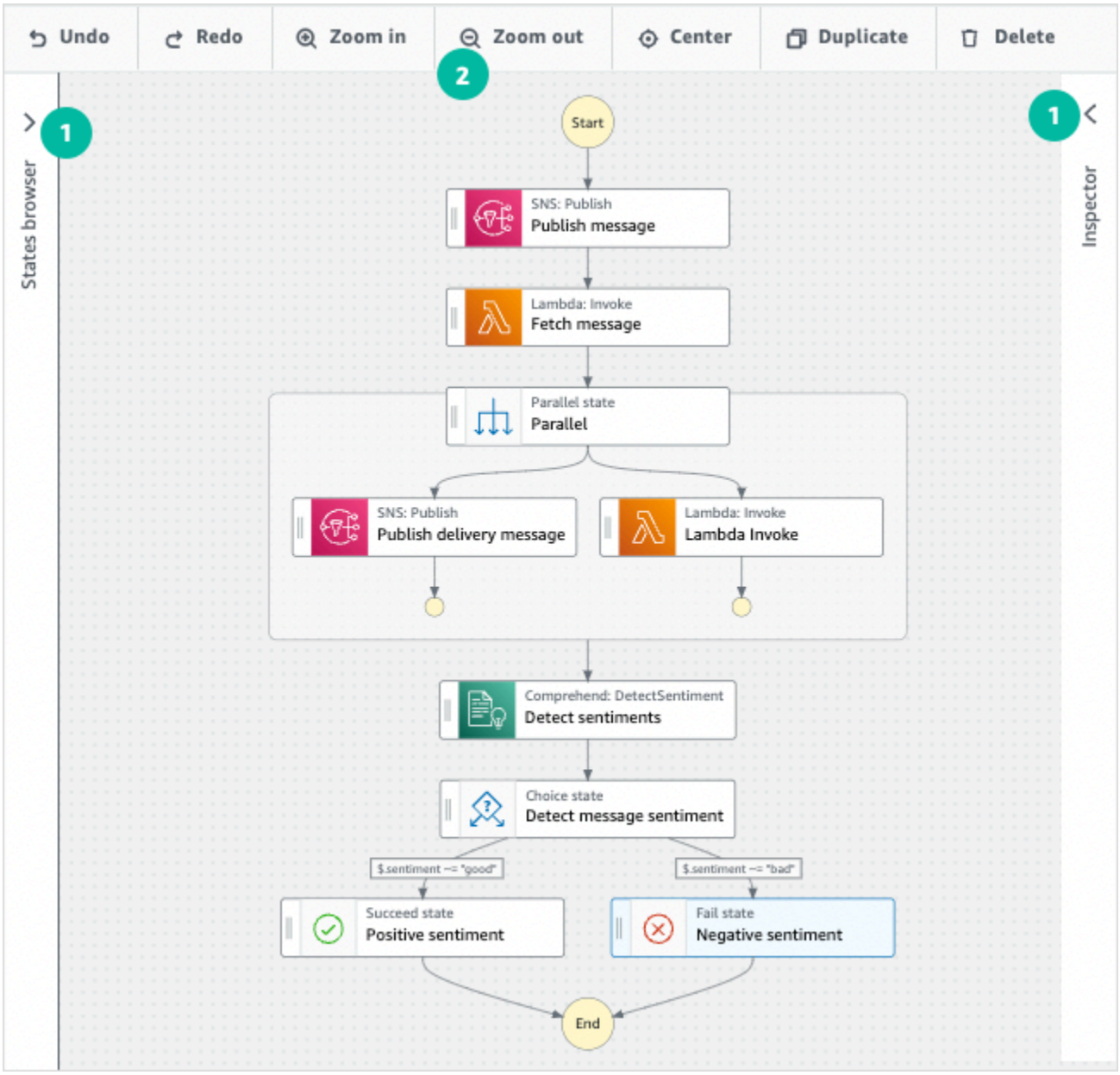
對於這些狀態，您可以使用適合您需求的組態來取代預留位置值。

若要刪除狀態，您可以使用退格鍵，按一下滑鼠右鍵並選擇「刪除」狀態，或選擇「設計」工具列上的「刪除」。



隨著工作流程的增長，它可能不適合在畫布上。您可以：

1. 使用側面板上的控制項，調整面板的大小或關閉面板。
2. 使用頂端的 [設計] 工具列控制項可放大或縮小工作流程圖表。Canvas



執行工作流程

使用工作流程 Studio 建立或編輯工作流程之後，您可以執行它，並在 Step Functions 式主控台中檢視其執行。

在工作流程工作室中執行工作流程

1. 在「設計」、「程式碼」或「Config」模式中，選擇「執行」。

   開始執行對話方塊會在新索引標籤中開啟。
2. 在 [開始執行] 對話方塊中，執行下列動作：

1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。
3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

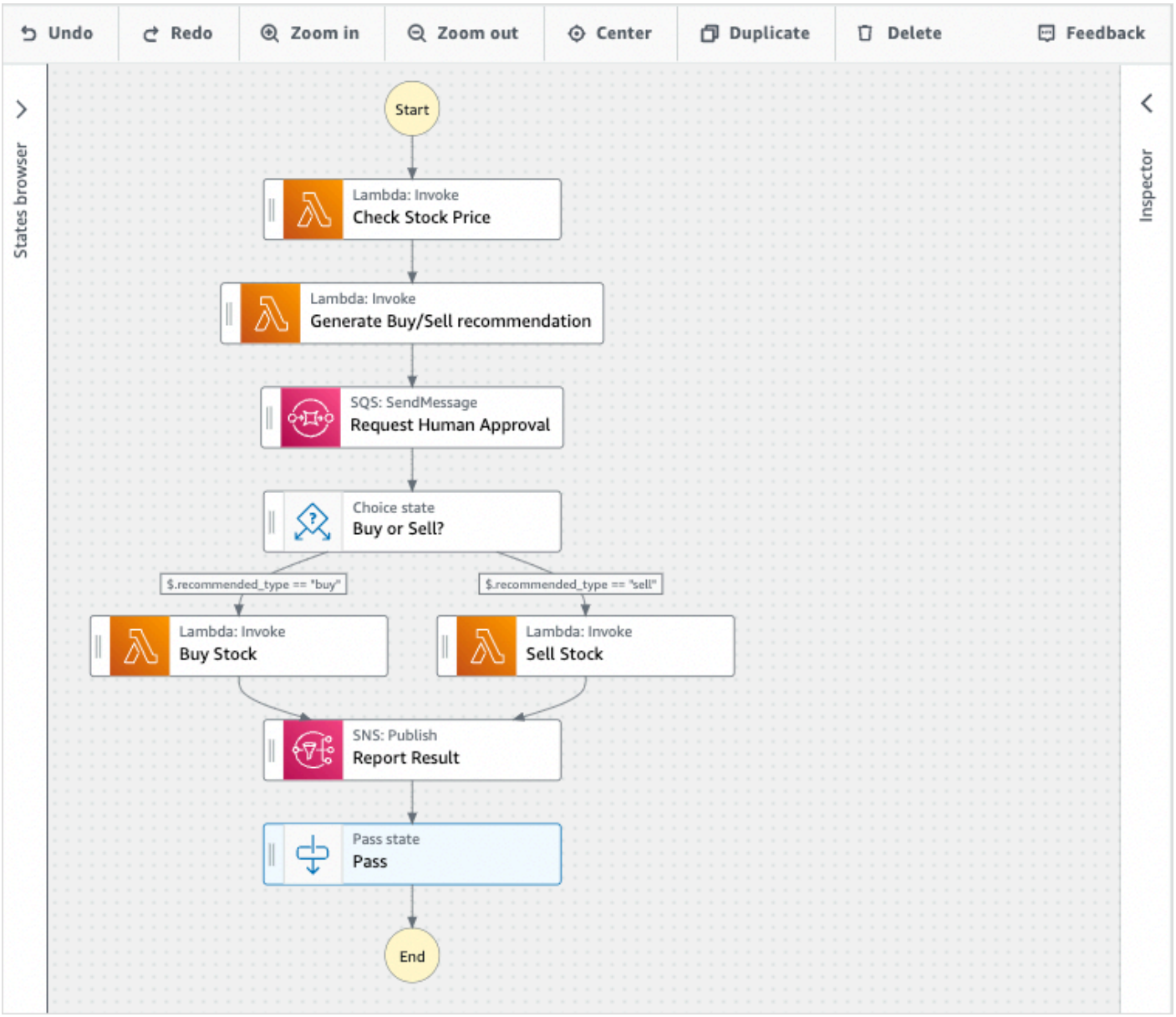
   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

編輯工作流程

您可以在工作流程 Studio 中以視覺方式編輯現有設計模式的工作流程。您也可以在工作流程 Studio 中編輯工程式碼模式作流程定義。

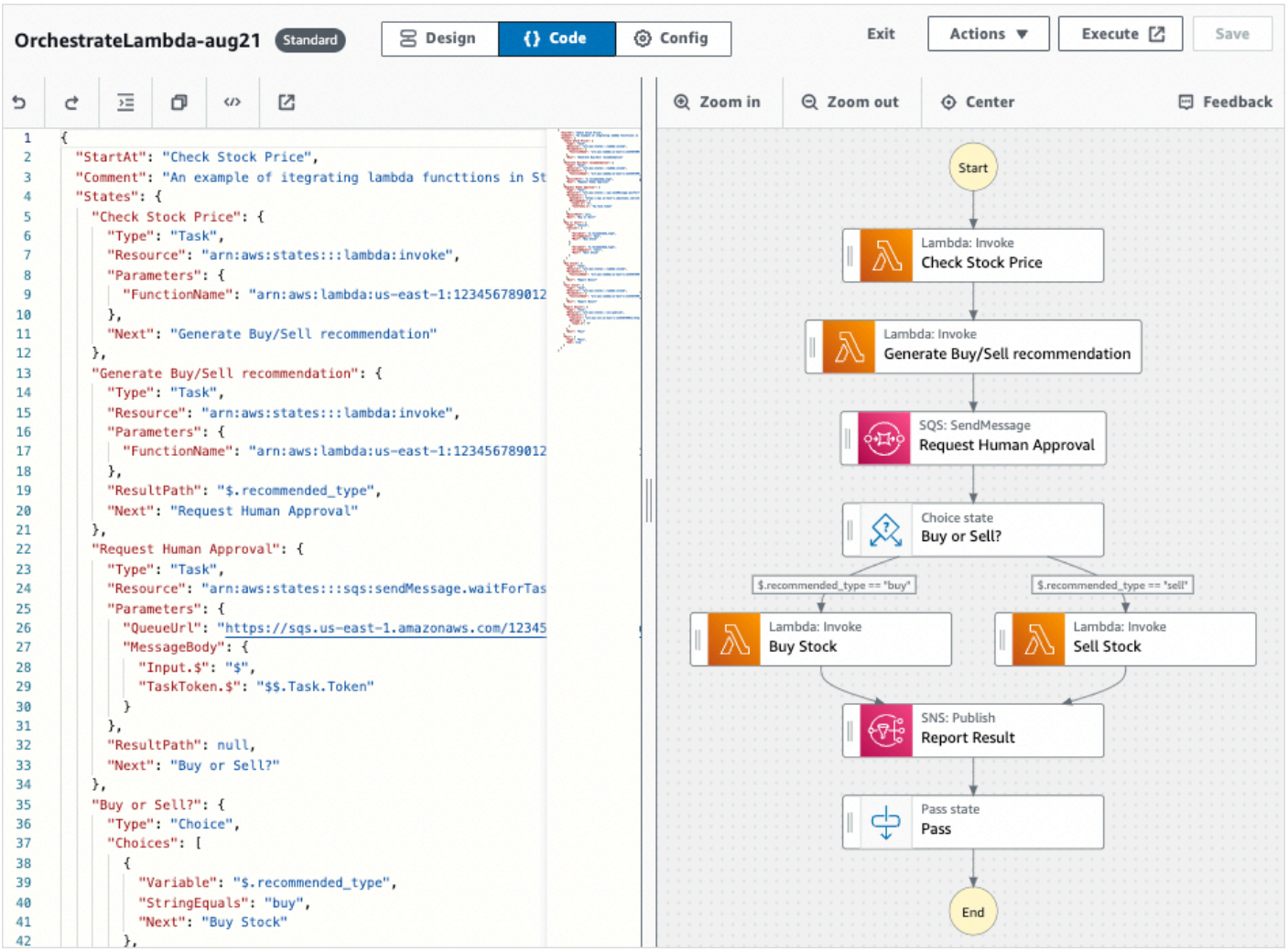
若要編輯現有的工作流程：

1. 開啟「Step Functions」主控台。
2. 在 [狀態機器] 頁面上，選擇您要編輯的工作流程。
3. 在狀態機器詳細資料頁面上，選擇編輯。
4. 工作流程會以工作流程 Studio 的 [設計] 模式開啟。視需要編輯工作流程。



> **Note**
>
> 如果您在工作流程中看到錯誤，則必須在「設計」模式中修正錯誤。如果工作流程中存在任何錯誤，則無法切換到「代碼」或「Config」模式。

5. (選擇性) 選擇 [程式碼] 按鈕以檢視或編輯工作流程 Studio 中的工作流程定義。



6. 完成後，選擇 [儲存] 以儲存更新的工作流程。
7. (選擇性) 若要執行更新的工作流程，請選擇「執行」。開始執行對話方塊會在新索引標籤中開啟。

匯出工作流程

您可以匯出工作流程的 Amazon States Language (ASL) 定義和工作流程圖表：

1. 在 Step Functions 主控台中選擇您的工作流程。
2. 在狀態機器詳細資料頁面上，選擇編輯。
3. (選擇性) 您的工作流程會在工作流程 Studio 的設計模式中開啟。在「設計」模式中編輯工作流程，或切換至「程式碼」模式。
4. 選擇「作業」 下拉式按鈕，然後執行下列其中一項或兩項作業：
   - 若要將工作流程圖形匯出為 SVG 或 PNG 檔案，請在「匯出圖形」下選取您要的格式。

- 若要將工作流程定義匯出為 JSON 或 YAML 檔案，請在 [匯出定義] 下選取您想要的格式。

建立工作流程原型

您可以使用工作流程 Studio 來建立包含預留位置資源的新工作流程原型。您也可以使用中的工作流程 Studio 來建置工作流程Application Composer。若要建立原型：

1. 登入 Step Functions 主控台。
2. 選擇 Create state machine (建立狀態機器)。
3. 在「選擇範本」對話方塊中，選取「空白」。
4. 選擇選取。這會在設計模式中開啟工作流程工作室
5. 工作流程工作室的設計模式隨即開啟。在工作流程工作室設計工作流程。若要包含預留位置資源：
   a. 選擇您要包含預留位置資源的狀態，然後在 [設定] 中：
      - 對於 Lambda 叫用狀態，請選擇函數名稱，然後選擇輸入函數名稱。您也可以輸入函數的自訂名稱。
      - 對於 Amazon SQS 傳送訊息狀態，請選擇佇列網址，然後選擇 [輸入佇列網址]。輸入預留位置佇列 URL。
      - 對於 Amazon SNS 發佈狀態，請從主題中選擇一個主題 ARN。
      - 對於「動作」 下列出的所有其他狀態，您可以使用預設組態。

      > **Note**
      > 如果您在工作流程中看到錯誤，則必須在「設計」模式中修正錯誤。如果工作流程中存在任何錯誤，則無法切換到「代碼」或「Config」模式。

   b. (選擇性) 若要檢視工作流程的自動產生 ASL 定義，請選擇「定義」。
   c. (選擇性) 若要更新工作流程 Studio 中的工作流程定義，請選擇 [程式碼] 按鈕。

      > **Note**
      > 如果您在工作流程定義中看到錯誤，則必須在「程式碼」模式中修正錯誤。如果工作流程定義中存在任何錯誤，則無法切換到「設計」或「Config 態」模式。

6. (選擇性) 若要編輯狀態機器名稱，請選擇預設狀態機器名稱旁的編輯圖示，MyStateMachine然後在 [狀態機器名稱] 方塊中指定名稱。

您也可以切換Config 模式至以編輯預設狀態機器名稱。

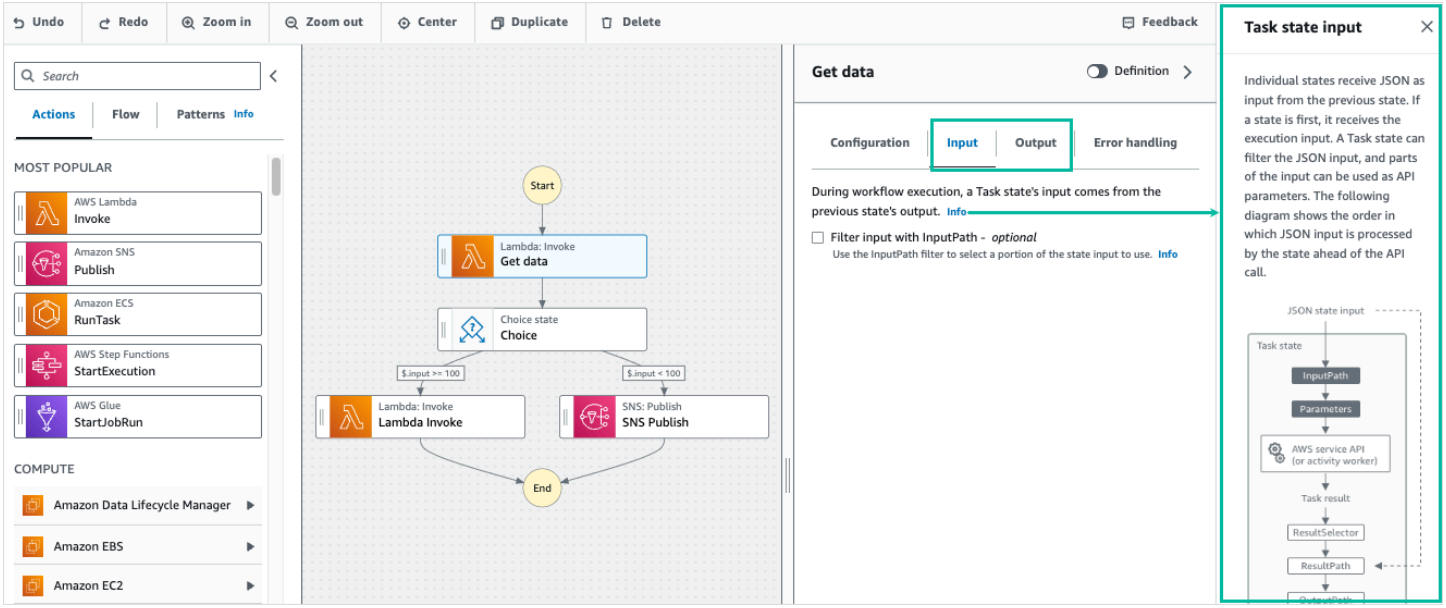
7. 指定您的工作流程設定，例如狀態機器類型及其執行角色。
8. 選擇建立。

您現在已建立新的工作流程，其中包含可用於製作原型的預留位置資源。您可以匯出工作流程定義和工作流程圖表。

- 若要將工作流程定義匯出為 JSON 或 YAML 檔案，請在 [設計] 或 [程式碼] 模式下，選擇 [動作] 下拉式按鈕。然後，在「匯出定義」下，選取您要匯出的格式。您可以使用此匯出的定義做為本機開發的起點AWS Toolkit for Visual Studio Code。
- 若要將工作流程圖形匯出為 SVG 或 PNG 檔案，請在「設計」或「程式碼」模式下，選擇「動作」下拉式按鈕。然後，在「匯出定義」下，選取所需的格式。

為您的狀態配置輸入和輸出

每個狀態都會根據收到的輸入做出決定或執行動作。在大多數情況下，它然後將輸出傳遞給其他狀態。在 Workflow Studio 中，您可以在Inspector面板的「輸入」和「輸出」索引標籤中設定狀態篩選和操作其輸入和輸出資料的方式。在設定輸入和輸出時，使用「資訊」連結來存取關聯式說明。



如需「Step Functions」如何處理輸入和輸出的詳細資訊，請參閱Step Functions 中的輸入和輸出處理。

設定狀態的輸入

每個狀態都會以 JSON 形式接收先前狀態的輸入。如果要過濾輸入，可以使用Inspector面板中「輸入」選項卡下的InputPath過濾器。這InputPath是一個字符串，開頭為$，標識特定的 JSON 節點。這些被稱為參考路徑，它們遵循 JsonPath 語法。

若要篩選輸入：

- 選擇篩選輸入方式 InputPath。
- 輸入一個有效JsonPath的過InputPath濾器。例如 **$.data**。

您的InputPath篩選器將新增至您的工作流程。

Example 範例 1：在工作流程工作室中使用 InputPath 篩選

假設輸入到您的狀態包含以下 JSON 數據。

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

若要套用InputPath濾鏡，請選擇「篩選輸入方式」 InputPath，然後輸入適當的參考路徑。如果您輸入**$.dataset2.val1**，則會將下列 JSON 做為狀態的輸入傳遞。

```
{"a"}
```

參考路徑也可以選取值。如果您參照的資料是，{ "a": [1, 2, 3, 4] }且您套用參照路徑$.a[0:2]作為InputPath篩選器，則結果如下。

```
[ 1, 2 ]
```

平行Map、和Pass流量狀態在其「輸入」索引標籤Parameters下有一個額外的輸入篩選選項。此篩選器會在 InputPath 篩選器之後生效，並可用來建構由一或多個索引鍵值配對組成的自訂 JSON 物件。每個配對的值可以是靜態值，可以從輸入中選取，也可以內容物件使用路徑從中選取。

> **Note**
>
> 若要指定參數使用參考路徑來指向輸入中的 JSON 節點，參數名稱必須以結尾.$。

Example 範例 2：為平行狀態建立自訂 JSON 輸入

假設下面的 JSON 數據是並行狀態的輸入。

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

若要選取此輸入的一部分並傳遞具有靜態值的其他鍵值配對，您可以在「平行」狀態的「輸入」標籤下的「參數」欄位中指定下列項目。

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

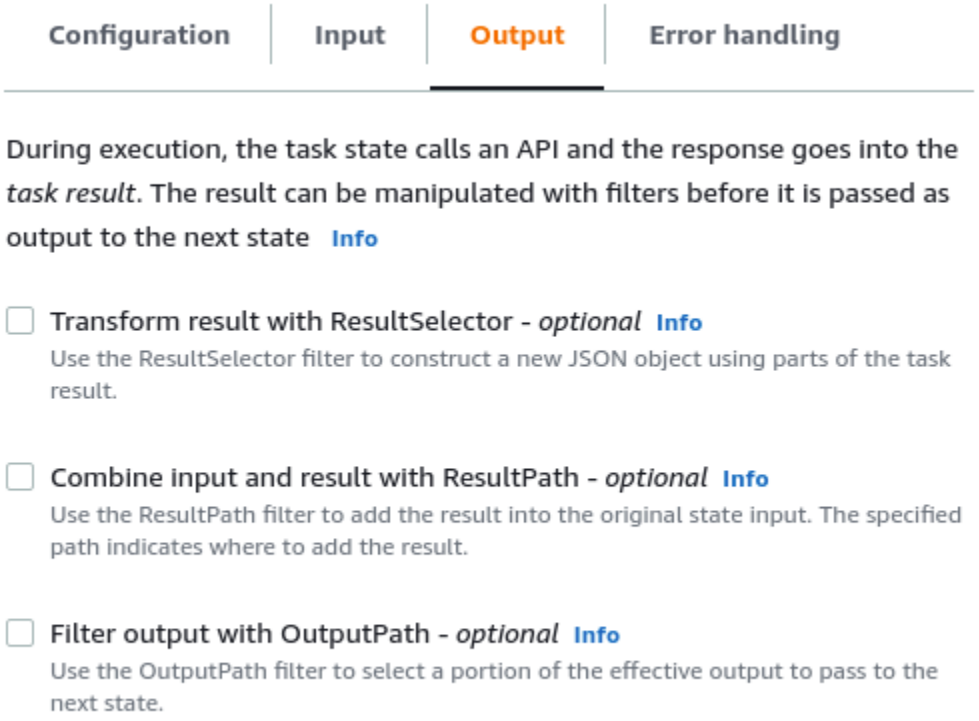
下面的 JSON 數據將是結果。

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

設定狀態的輸出

每個狀態都會產生 JSON 輸出，這些輸出可以在傳遞到下一個狀態之前進行篩選。有多個濾鏡可用，每個濾鏡都會以不同的方式影響輸出。每個狀態可用的輸出濾鏡會列在「Inspector 器」面板的「輸出」索引標籤下。對於任務狀態，您選取的任何輸出篩選器都會以下列順序處理：

1. `ResultSelector`：使用此篩選器來操作狀態的結果。您可以使用部分結果構建一個新的 JSON 對象。
2. `ResultPath`：使用此篩選器可選取要傳遞至輸出的狀態輸入和作業結果的組合。
3. `OutputPath`：使用此篩選器篩選 JSON 輸出，以選擇將結果中的哪些資訊傳遞至下一個狀態。



使用 ResultSelector

`ResultSelector`是下列狀態的選用輸出篩選器：

- 任務狀態，這些狀態是列在的「動作」標籤中的所有狀態狀態瀏覽器。
- Map狀態，在「狀態」瀏覽器的「流量」索引標籤中。
- 平行狀態，在「狀態」瀏覽器的「流量」索引標籤中。

`ResultSelector`可用來建構由一或多個索引鍵值配對組成的自訂 JSON 物件。每個配對的值可以是靜態值，也可以是從狀態的結果中選取路徑。

> **Note**
>
> 若要指定參數使用路徑來參照結果中的 JSON 節點，參數名稱必須以結尾`.$`。

Example 使用 ResultSelector 過濾器的示例

在此範例中，您可ResultSelector以使用來操作來自 Amazon EMR 狀態之 Amazon EMR CreateCluster API 呼叫的回應。CreateCluster以下是 Amazon EMR CreateCluster API 調用的結果。

```
{
  "resourceType": "elasticmapreduce",
  "resource": "createCluster.sync",
  "output": {
    "SdkHttpMetadata": {
      "HttpHeaders": {
        "Content-Length": "1112",
        "Content-Type": "application/x-amz-JSON-1.1",
        "Date": "Mon, 25 Nov 2019 19:41:29 GMT",
        "x-amzn-RequestId": "1234-5678-9012"
      },
      "HttpStatusCode": 200
    },
    "SdkResponseMetadata": {
      "RequestId": "1234-5678-9012"
    },
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

若要選取部分此資訊，並傳遞含有靜態值的其他鍵值組，請在狀態的「輸出」(Output) 標籤下的ResultSelector欄位中指定下列內容。

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

使用ResultSelector產生以下結果。

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
 "ResourceType": "elasticmapreduce"
}
```

使用 ResultPath

狀態的輸出可以是其輸入的副本，它產生的結果，或者它的輸入和結果的組合。使用 ResultPath 以控制哪些組合會傳遞至狀態輸出。如需的更多使用案例ResultPath，請參閱ResultPath。

ResultPath是下列狀態的選用輸出篩選器：

- 任務狀態，這些狀態是列在「狀態」瀏覽器的「動作」標籤中的所有狀態。
- Map狀態，在「狀態」瀏覽器的「流量」索引標籤中。
- 平行狀態，在「狀態」瀏覽器的「流量」索引標籤中。
- Pass狀態，在「狀態」瀏覽器的「流量」索引標籤中。

ResultPath可用於將結果添加到原始狀態輸入中。指定的路徑指示在何處添加結果。

Example 使用 ResultPath 過濾器的示例

假設以下是任務狀態的輸入。

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

「工作」狀態的結果如下。

```
Hello, AWS Step Functions
```

您可以套用ResultPath並輸入指出要新增結果位置的參考路徑，將此結果新增至狀態的輸入中，例如$.taskresult：

這樣ResultPath，以下是作為狀態輸出傳遞的 JSON。

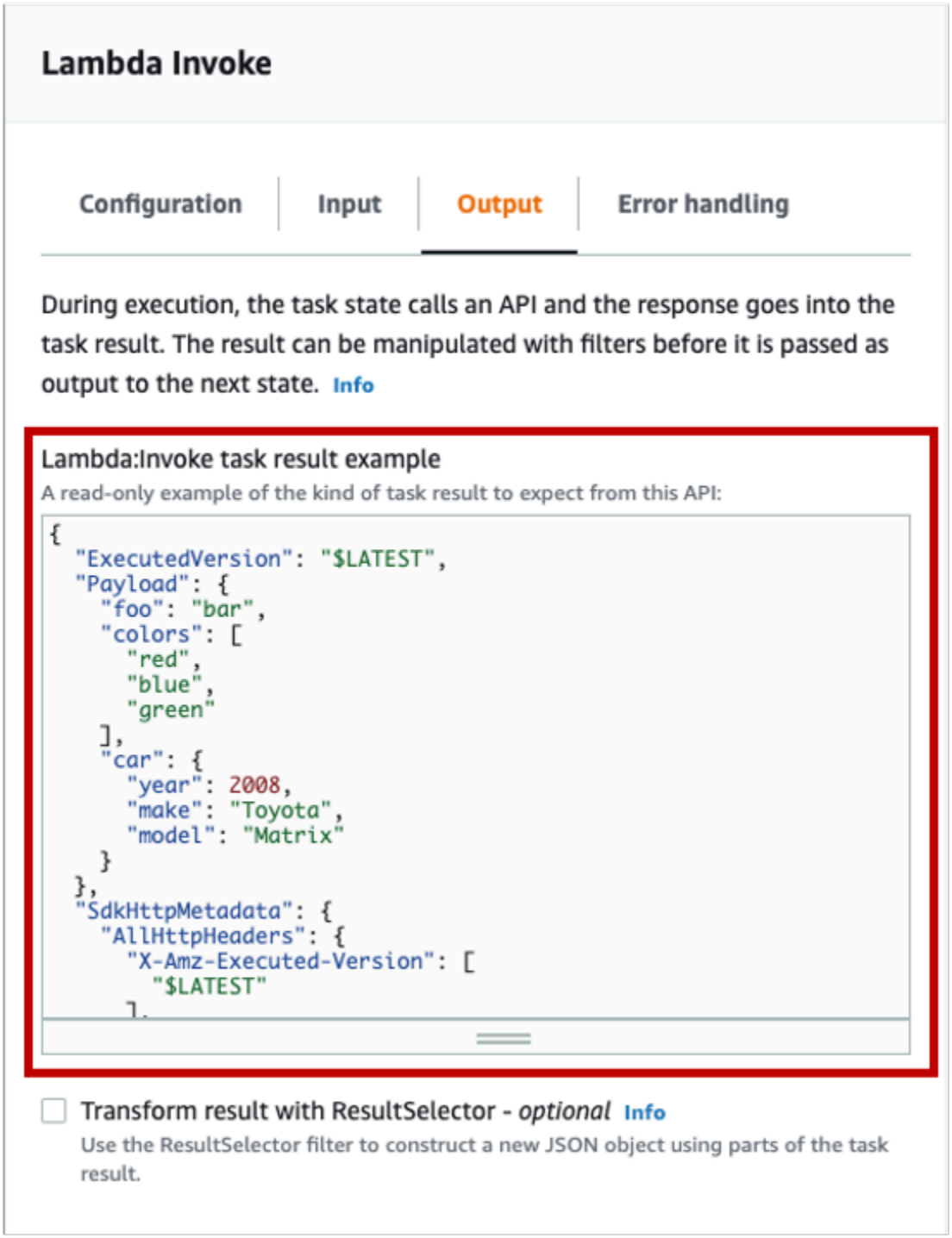
```
{
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

使用 OutputPath

該OutputPath過濾器可以讓您過濾掉不需要的信息，並僅傳遞您關心的 JSON 部分。
這OutputPath是一個字符串$，用於標識 JSON 文本中的節點。

Example 使用 OutputPath 過濾器的示例

除了有效負載之外，Lambda 叫用 API 呼叫還會傳回中繼資料，也就是 Lambda 函數的結果。此 API 呼叫的回應範例會顯示在狀態的 [輸出] 索引標籤下。



您可以使用`OutputPath`來篩選出其他中繼資料。根據預設，透過工作流程 Studio 建立的 Lambda 叫用狀態的OutputPath篩選器值為`$.Payload`。此預設值會移除其他中繼資料，並傳回等同於直接執行 Lambda 函數的輸出。

Lambda 叫用工作結果範例和輸出篩選器的`$.Payload`值會傳遞下列 JSON 資料做為輸出。

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> 由於OutputPath濾鏡是最後一個生效的輸出濾鏡，因此如果您使用其他輸出濾鏡 (例如ResultSelector或)ResultPath，您應該相應地修改OutputPath濾鏡的$.Payload預設值。

工作流程中的執行角色

每個Step Functions狀態機器都需要一個 AWS Identity and Access Management （IAM）角色，該角色授予狀態機器對資源執行 AWS 服務 操作或調用第三方 API 的權限。此角色稱為執行角色。此角色必須包含IAM每個動作的政策，例如，允許狀態機調用AWS Lambda函數，運行AWS Batch作業或調用 Stripe API 的策略。 Step Functions在下列情況下，會要求您提供執行角色：

- 您可以在主控台、 AWS SDK 或 AWS CLI 使用 CreateStateMachineAPI 中建立狀態機器。
- 您可以在主控台、 AWS SDK 或AWS CLI使用 TestStateAPI 中測試狀態。

工作流程 Studio 的功能可讓您輕鬆管理工作流程的執行角色。

主題

- 關於產生的角色
- 自動產生角色
- 解決角色產生問題
- 在工作流程工作室中測試 HTTP 任務的角色

- 在工作流程工作室中測試優化服務集成的角色
- 在工作流工作室中測試 AWS SDK 服務整合的角色
- 在工作流程 Studio 中測試流程狀態的角色

關於產生的角色

當您在Step Functions主控台中建立狀態機器時，Workflow Studio 可以自動為您建立包含必要IAM原則的執行角色。Workflow Studio 會分析您的狀態機器定義，並以執行工作流程所需的最少權限來產生原則。

工作流程 Studio 可以產生下列項目的IAM原則：

- 呼叫第三方 API 的 HTTP 工作。
- AWS 服務 使用最佳化整合 (例如 Invoke、或) Lambda呼叫其他的工作狀態AWS Glue StartJobRun。DynamoDB GetItem
- 執行巢狀工作流程的工作狀態。
- 分散式地圖狀態，包括啟動子工作流程執行、列出儲存Amazon S3貯體以及讀取或寫入 S3 物件的政策。
- X-Ray追蹤。Workflow Studio 中自動生成的每個角色都包含一個策略，該策略授予狀態機器將跟踪發送到的權限X-Ray。
- 記錄使用CloudWatch日誌在狀態機上啟用日誌記錄時。

工作流 Studio 無法生成調 AWS 服務 用其他使用 AWS SDK 集成的任務狀態IAM策略。

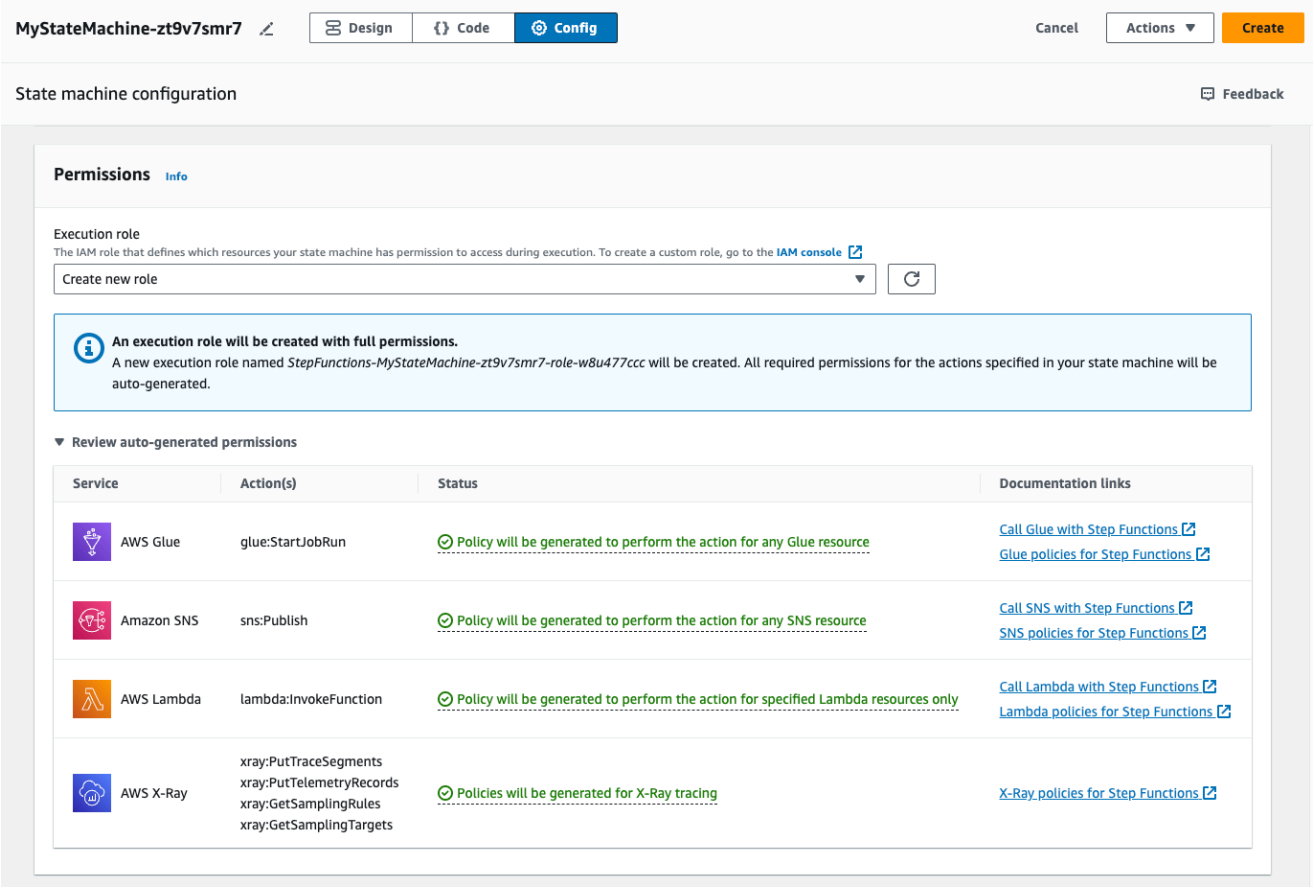
自動產生角色

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。

   您也可以更新現有的狀態機器。如果您要更新狀態機，請參閱步驟 4。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 選擇「Con fig」頁標。
5. 向下捲動至「權限」區段，然後執行下列動作：

   a. 對於「執行」角色，請確保保留「創建新角色」的默認選擇。

工作流程 Studio 會針對狀態機器定義中的每個有效狀態，自動產生所有必要的IAM原則。它顯示一個橫幅與消息，執行角色將創建具有完整權限。



Tip

若要檢閱 Workflow Studio 為狀態機器自動產生的權限，請選擇 [檢閱自動產生的權限]。

Note

如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

如果 Workflow Studio 無法產生所有必要的IAM原則，它會顯示一個橫幅，其中包含無法自動產生某些動作的權限訊息。將僅使用部分權限建立IAM角色。如需如何新增遺失權限的詳細資訊，請參閱解決角色產生問題。

b. 如果您正在建立狀態機，請選擇 [建立]。否則，請選擇 Save (儲存)。

c. 在出現的對話方塊中選擇「確認」。

工作流 Studio 會儲存您的狀態機器，並建立新的執行角色。

解決角色產生問題

在下列情況下，工作流程 Studio 無法自動產生具有所有必要權限的執行角色：

- 您的狀態機器中有錯誤。請務必解決工作流程工作室中的所有驗證錯誤。另外，請確保您解決了在保存過程中遇到的任何服務器端錯誤。
- 您的狀態機包含使用 AWS SDK 集成的任務。在此情況下，工作流程 Studio 無法自動產生IAM原則。工作流 Studio 顯示帶有消息的橫幅，某些操作的權限無法自動生成。將僅使用部分權限建立IAM角色。在「檢閱自動產生的權限」表格中，選擇「狀態」 中的內容，以取得有關遺失執行角色之原則的詳細資訊。工作流程 Studio 仍然可以產生執行角色，但這個角色不會包含所有動作的IAM原則。請參閱說明文件連結下的連結，以撰寫您自己的原則，並在產生原則後將其新增至角色。即使在儲存狀態機器之後，這些連結仍可使用。

在工作流程工作室中測試 HTTP 任務的角色

您需要執行角色才能測試 HTTP 任務狀態。如果您沒有具有足夠權限的角色，請使用下列其中一個選項來建立角色：

- 使用工作流程 Studio 自動生成角色（推薦）-這是安全的選項。關閉「測試狀態」對話方塊，並按照中的指示進行操作自動產生角色。這將需要您先創建或更新狀態機，然後返回到 Workflow Studio 來測試您的狀態。
- 使用具有管理員存取權限的角色 — 如果您具有建立角色的權限，可以對中的所有服務和資源具有完整存取權限 AWS，則可以使用該角色來測試工作流程中任何類型的狀態。若要這麼做，您可以在IAM主控台中建立Step Functions服務角色，並將AdministratorAccess 原則新增至該角色 https://console.aws.amazon.com/iam/。

在工作流程工作室中測試優化服務集成的角色

您需要執行角色才能呼叫最佳化服務整合的 Task 狀態。如果您沒有具有足夠權限的角色，請使用下列其中一個選項來建立角色：

- 使用 Workflow Studio 中的文件連結來撰寫您自己的IAM原則 (建議使用) — 這是安全選項。關閉「測試狀態」對話方塊，並按照中的指示進行操作自動產生角色。這將需要您先創建或更新狀態機，然後返回到 Workflow Studio 來測試您的狀態。
- 使用具有管理員存取權限的角色 — 如果您具有建立角色的權限，可以對中的所有服務和資源具有完整存取權限 AWS，則可以使用該角色來測試工作流程中任何類型的狀態。若要這麼做，您可以在IAM主控台中建立Step Functions服務角色，並將AdministratorAccess 原則新增至該角色 https://console.aws.amazon.com/iam/。

在工作流工作室中測試 AWS SDK 服務整合的角色

您需要調AWS 用 SDK 集成的任務狀態的執行角色。如果您沒有具有足夠權限的角色，請使用下列其中一個選項來建立角色：

- 使用 Workflow Studio 中的文件連結來撰寫您自己的IAM原則 (建議使用) — 這是安全選項。關閉「測試狀態」對話方塊，並按照中的指示進行操作自動產生角色。這將需要您先創建或更新狀態機，然後返回到 Workflow Studio 來測試您的狀態。請執行下列操作：
  1. 關閉「測試狀態」對話方塊
  2. 選擇「Con fig」頁籤以檢視「Config」模式。
  3. 向下捲動至「權限」區段。
  4. 工作流 Studio 顯示帶有消息的橫幅，某些操作的權限無法自動生成。將僅使用部分權限建立IAM角色。選擇檢閱自動產生的權限
  5. 「檢閱自動產生的權限」表格會顯示一列，其中顯示與您要測試之工作狀態對應的動作。請參閱文件連結下的連結，將您自己的IAM原則寫入自訂角色。
- 使用具有管理員存取權限的角色 — 如果您具有建立角色的權限，可以對中的所有服務和資源具有完整存取權限 AWS，則可以使用該角色來測試工作流程中任何類型的狀態。若要這麼做，您可以在IAM主控台中建立Step Functions服務角色，並將AdministratorAccess 原則新增至該角色 https://console.aws.amazon.com/iam/。

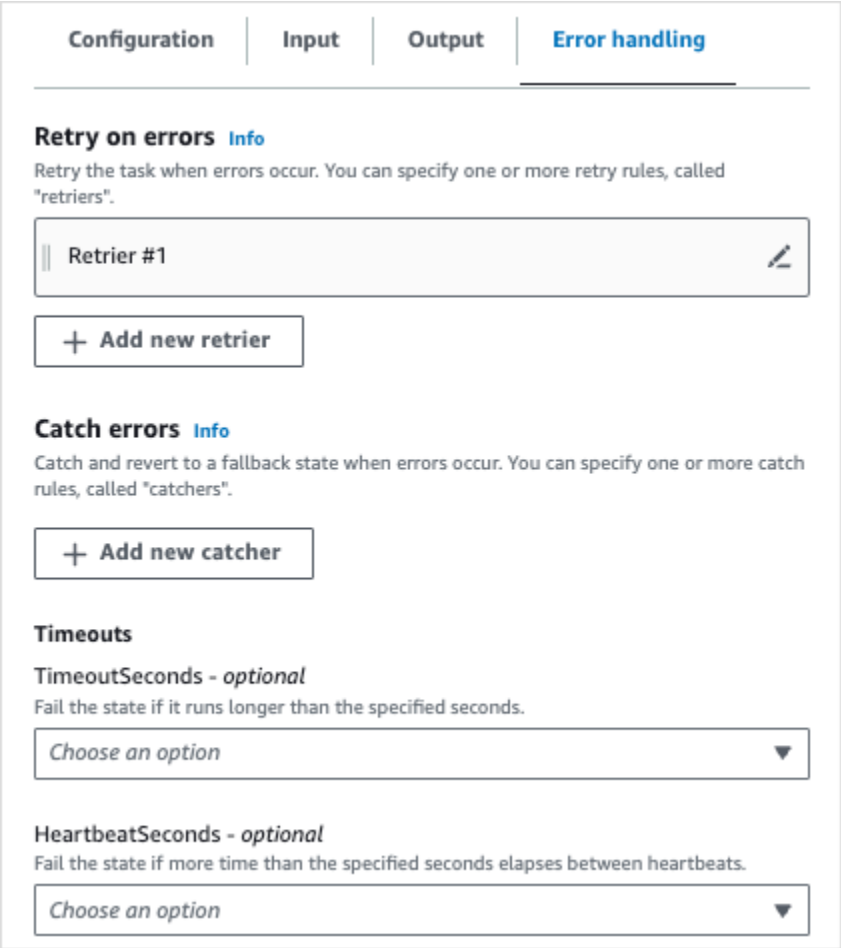
在工作流程 Studio 中測試流程狀態的角色

您需要執行角色，才能在工作流程 Studio 中測試流程狀態。流程狀態是指直接執行流程的狀態Choice平行，例如MapPass、等候、、Succeed、或Fail。TestStateAPI 不適用於 [地圖] 或 [平行] 狀態。使用下列其中一個選項建立用於測試流程狀態的角色：

- 使用 AWS 帳戶 (建議) 中的任何角色 — 流程狀態不需要任何特定IAM原則，因為它們不會呼叫 AWS 動作或資源。因此，您可以IAM在 AWS 帳戶.
  1. 在 [測試狀態] 對話方塊中，從 [執行角色] 下拉式清單中選取任何角色。
  2. 如果下拉式清單中沒有出現任何角色，請執行下列動作：
     a. 在IAM主控台 https://console.aws.amazon.com/iam/ 中，選擇「角色」。
     b. 從清單中選擇角色，然後從角色詳細資訊頁面複製其 ARN。您將需要在「測試狀態」對話方塊中提供此 ARN。
     c. 在 [測試狀態] 對話方塊中，從 [執行角色] 下拉式清單中選取 [輸入角色 ARN]。
     d. 在角色 ARN 中粘貼 AR N。
- 使用具有管理員存取權限的角色 — 如果您具有建立角色的權限，可以對中的所有服務和資源具有完整存取權限 AWS，則可以使用該角色來測試工作流程中任何類型的狀態。若要這麼做，您可以在IAM主控台中建立Step Functions服務角色，並將AdministratorAccess 原則新增至該角色 https://console.aws.amazon.com/iam/。

錯誤處理

依預設，當狀態報告錯誤時，「Step Functions」會導致工作流程執行完全失敗。對於動作和某些流程狀態，您可以設定 Step Functions 如何處理錯誤。即使您已設定錯誤處理，某些錯誤仍可能導致工作流程執行失敗。如需詳細資訊，請參閱 Step Functions 中的錯誤處理。在工作流程 Studio 中，在Inspector面板的 [錯誤處理] 索引標籤中設定錯誤處理。



發生錯誤時重試

您可以將一或多個規則新增至動作狀態，並在發生錯誤時重試工作的平行流程狀態。這些規則稱為擷取器。若要新增擷取器，請在「擷取器 #1」方塊中選擇編輯圖示，然後設定其選項：

- (選擇性) 在「註解」欄位中，新增您的註解。它不會影響工作流程，但可用於註釋您的工作流程。
- 將游標置於「錯誤」欄位中，然後選擇會觸發擷取器的錯誤，或輸入自訂錯誤名稱。您可以選擇或新增多個錯誤。
- (選擇性) 設定間隔。這是 Step Functions 進行第一次重試之前的時間（以秒為單位）。額外的重試次數將按照您可以使用最大嘗試次數和輪詢率進行配置的間隔。
- (選擇性) 設定最大嘗試次數。這是 Step Functions 數導致執行失敗之前的最大重試次數。
- (選擇性) 設定輪詢率。這是由每次嘗試增加重試間隔的倍數來決定。

> **Note**
>
> 並非所有的錯誤處理選項都適用於所有狀態。Lambda 叫用預設會設定一個擷取器。

捕捉錯誤

您可以將一或多個規則新增至動作狀態以平行及和Map流程狀態，以 catch 錯誤。這些規則稱為捕手。若要新增捕捉器，請選擇「新增捕手」，然後設定其選項：

- (選擇性) 在「註解」欄位中，新增您的註解。它不會影響工作流程，但可用於註釋您的工作流程。
- 將游標置於「錯誤」欄位中，然後選擇會觸發捕捉器的錯誤，或輸入自訂錯誤名稱。您可以選擇或新增多個錯誤。
- 在「後援狀態」欄位中，選擇後援狀態。這是工作流程將移至下一個狀態，在發現錯誤之後。
- (選擇性) 在ResultPath欄位中，新增`ResultPath`篩選器，以將錯誤新增至原始狀態輸入。必`ResultPath`須是有效的JsonPath。這將會傳送至後援狀態。

逾時

您可以設定動作狀態逾時，以設定狀態失敗前可執行的秒數上限。使用超時來防止卡住的執行。若要設定逾時，請輸入狀態在執行失敗之前應等待的秒數。如需逾時的詳細資訊，請參閱任務狀態`TimeoutSeconds`中。

HeartbeatSeconds

您可以設定工作傳送的活動訊號或定期通知。如果您設定活動訊號間隔，且您的狀態不會以設定的間隔傳送活動訊號通知，則工作會標記為失敗。若要設定活動訊號，請設定正、非零的整數秒數。如需詳細資訊，請參閱任務狀態`HeartBeatSeconds`中。

自學課程：學習使用 AWS Step Functions 工作流程工作室

在本自學課程中，您將學習使用的工作流程 Studio 的基本知識 AWS Step Functions。在工設計模式作流程 Studio 中，您將創建一個包含多個狀態的狀態機器 `PassChoice`，包括`Fail`，`Wait`，，和`Parallel`。您將使用拖放功能來搜尋、選取和設定這些狀態。然後，您將檢視工作流程的自動產生Amazon States Language (ASL) 定義。您還將使用程式碼模式的工作流 Studio 來編輯工作流定義。然後，您將結束工作流程 Studio，執行狀態機器，並檢閱執行詳細資料。

在本教程中，您還將學習如何更新狀態機器並查看執行輸出中的更改。最後，您將執行清理步驟並刪除狀態機器。

完成這個教學課程之後，您就會知道如何使用 Workflow Studio 來建立和設定使用 [設計] 和 [程式碼] 模式的工作流程。您還將知道如何更新，運行和刪除狀態機。

> **Note**
>
> 開始之前，請確定已完成本教學課程的先決條件。

主題

步驟 1：瀏覽至工作流程工作室

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室

步驟 2：建立狀態機

在工作流程 Studio 中，狀態機器是工作流程的圖形表示。使用工作流程 Studio，您可以定義、設定和檢查工作流程的個別步驟。在下面的步驟中，您可以使用設計模式的工作流 Studio 來創建您的狀態機。

建立 狀態機器

1. 請確定您處於 [工作流程工作室] 的 [設計] 模式。

2. 從左狀態瀏覽器側選擇「流量」頁標。然後，將「通過」狀態拖曳至標示為「拖曳第一個狀態」的空白狀態。
3. 將「選擇」狀態從「流程」標籤拖放到「通過」狀態下方。
4. 對於「狀態」名稱，請取代「選擇」的預設名稱。在本教學課程中，使用名稱 **IsHelloWorldExample**。
5. 拖曳另一個「通過」狀態並將其拖放到IsHelloWorldExample狀態的一個分支。然後，將「失敗」狀態拖放到狀IsHelloWorldExample態的其他分支下方。
6. 選擇「通過 (1)」狀態，並將其重新命名為**Yes**。將「失敗」狀態重新命名為**No**。
7. 使用布林變數`IsHelloWorldExample`指定IsHelloWorldExample狀態的分支邏輯。

   如果`IsHelloWorldExample`是`False`，則工作流程將進入「否」狀態。否則，工作流程會繼續處於 「是」 狀態的執行流程。

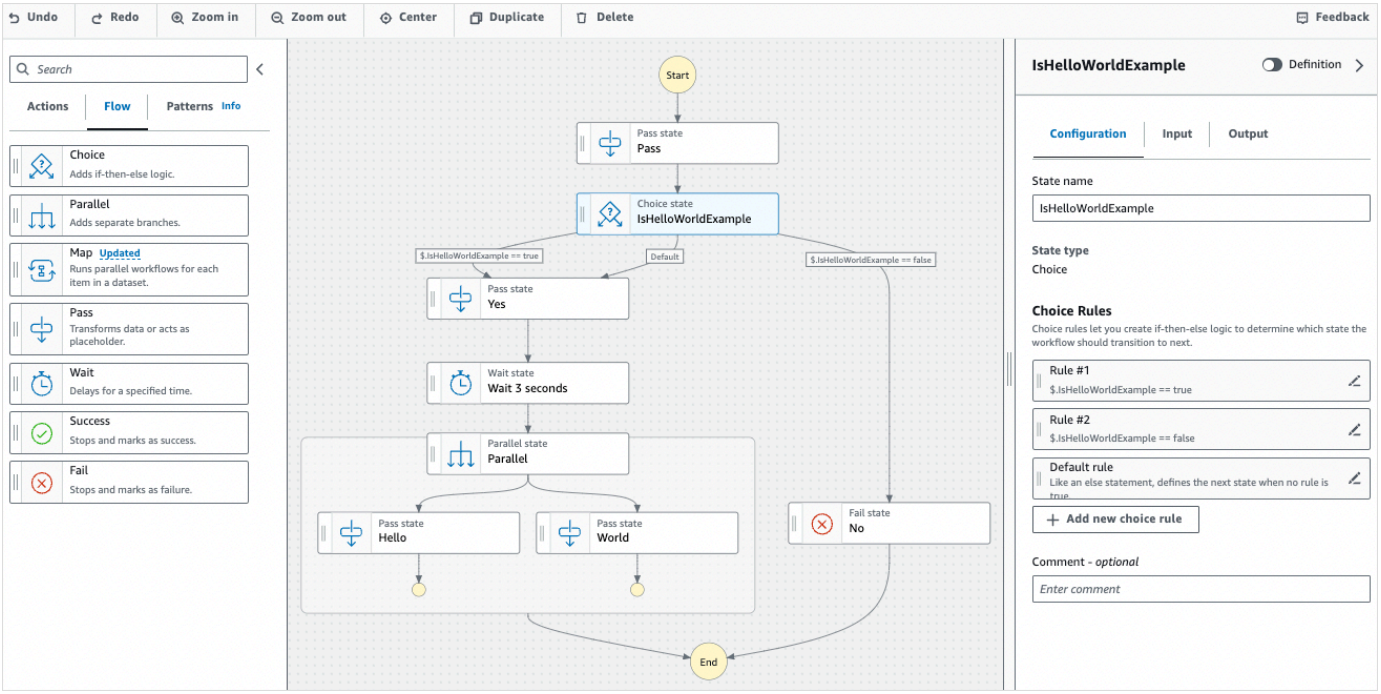
   若要定義分支邏輯，請執行下列操作：

   a. 在上選擇IsHelloWorldExample狀態Canvas，然後在「選擇規則」下，選擇「規則 #1」方塊中的編輯圖示，以定義第一個選擇規則。
   b. 選擇新增條件。
   c. 在「規則 #1 的條件」對話方塊中，於「變數」**$.IsHelloWorldExample** 下輸入。
   d. 選擇等於下運算符。
   e. 選擇布爾常量下值，然後從下拉列表中選擇 true。
   f. 選擇 [儲存條件]。
   g. 確保然後下一個狀態是：下拉列表已選擇是。
   h. 選擇「新增選擇規則」，然後選擇「新增條件」。
   i. 在「規則 #2」方塊中，透過重複子步驟 7.c 到 7.f 來定義`IsHelloWorldExample`變數值為 false 時的第二個選擇規則。對於步驟 7.e，選擇假而不是真。
   j. 在裡面規則 #2 框中，選擇否來自然後下一個狀態是：下拉列表。
   k. 在 [預設規則] 方塊中，選擇編輯圖示以定義預設選擇規則，然後從下拉式清單中選擇 [是]。
8. 在 [是] 狀態之後新增 [等待] 狀態，並將其命名**Wait 3 sec**。然後，通過執行以下步驟將等待時間配置為三秒鐘：

   a. 在 [選項] 下，保留 [等待固定間隔] 的預設選取項目。
   b. 在 [秒] 下，確定已選取 [輸入秒數]，然後**3**在方塊中輸入。

9. 在「等待 3 秒」狀態之後，新增「平行」狀態。在其兩個分支中添加兩個「通過」狀態。命名第一個「通過」狀態**Hello**。命名第二個「通過」狀態**World**。

   完成的工作流程如下所示：



步驟 3：檢閱自動產生的 Amazon 州語言定義

當您將狀態從「流程」索引標籤拖放到畫布上時，工作流程 Studio 會自動即時編寫工作流程的 Amazon 州語言 (ASL) 定義。在面Inspector板中，選擇「定義」切換按鈕以檢視此定義，或視需要切換程式碼模式至編輯此定義。若要取得有關編輯工作流程定義的資訊，請參閱本自學課程的步驟 4。

- (選擇性) 在「Inspector 查程式」面板上選擇「定義」，然後檢視狀態機的工作流程。

  下列範例程式碼顯示`IsHelloWorldExample`狀態機器自動產生的 Amazon 狀態語言定義。您在工作流程 Studio 中新增的`Choice`狀態會根據您在步驟 2 中定義的分支邏輯來決定執行流程。

```
{
  "Comment": "A Hello World example of the Amazon States Language using Pass
 states",
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "IsHelloWorldExample",
```

```
    "Comment": "A Pass state passes its input to its output, without performing
work. Pass states are useful when constructing and debugging state machines."
  },
  "IsHelloWorldExample": {
    "Type": "Choice",
    "Comment": "A Choice state adds branching logic to a state machine. Choice
rules can implement 16 different comparison operators, and can be combined using
And, Or, and Not\"",
    "Choices": [
      {
        "Variable": "$.IsHelloWorldExample",
        "BooleanEquals": false,
        "Next": "No"
      },
      {
        "Variable": "$.IsHelloWorldExample",
        "BooleanEquals": true,
        "Next": "Yes"
      }
    ],
    "Default": "Yes"
  },
  "No": {
    "Type": "Fail",
    "Cause": "Not Hello World"
  },
  "Yes": {
    "Type": "Pass",
    "Next": "Wait 3 sec"
  },
  "Wait 3 sec": {
    "Type": "Wait",
    "Seconds": 3,
    "Next": "Parallel"
  },
  "Parallel": {
    "Type": "Parallel",
    "End": true,
    "Branches": [
      {
        "StartAt": "Hello",
        "States": {
          "Hello": {
            "Type": "Pass",
```

```
              "End": true
            }
          }
        },
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

步驟 4：在「程式碼」模式中編輯工作流程定義

工作流程 Studio 的程式碼模式提供整合的程式碼編輯器，以檢視和編輯工作流程的 ASL 定義。

1. 選擇「程式碼」以切換至「程式碼」模式。
2. 在「平行」狀態定義之後，放置游標並按下**Enter**。
3. 按下**Ctrl+space**以查看您可以在「平行」狀態之後新增的狀態清單。
4. 從選項清單中選擇「通過狀態」。程式碼編輯器會新增「通過狀態」的樣板程式碼。
5. 新增此狀態會導致工作流程定義中發生錯誤。在「平行」狀態的定義中，取代`"End": true`為**`"Next": "PassState"`**。
6. 在您新增的「通過狀態」定義中，進行下列變更：

   a. 移除「結果」節點。

   b. 移除`"ResultPath": "$.result",`和`"Next": "NextState"`。

   c. 之後`"Type": "Pass",`，輸入**`"End": true`**。

   d. **`,`**在「通過狀態」定義之後新增 a。

您的工作流程定義現在看起來應該類似於下列定義。

```
{
```

```
  "Comment": "A description of my state machine",
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "Yes"
        },
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": false,
          "Next": "No"
        }
      ],
      "Default": "Yes"
    },
    "Yes": {
      "Type": "Pass",
      "Next": "Wait 3 seconds"
    },
    "Wait 3 seconds": {
      "Type": "Wait",
      "Seconds": 3,
      "Next": "Parallel"
    },
    "Parallel": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Hello",
          "States": {
            "Hello": {
              "Type": "Pass",
              "End": true
            }
          }
        },
```

```
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      ],
      "Next": "PassState"
    },
    "PassState": {
      "Type": "Pass",
      "End": true
    },
    "No": {
      "Type": "Fail"
    }
  }
}
```

步驟 5：保存狀態機

1. 選擇其他 Con fig 態或選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。在狀態機器組態中，指定名稱。例如，輸入 **HelloWorld**。
2. (選擇性) 指定其他工作流程設定，例如狀態機器類型及其執行角色。在本教學課程中，請保留狀態機器組態中的所有預設選項。
3. 選擇建立。
4. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色組態] 回到 [設定] 模式。

如需有關 Config 模式的詳細資訊，請參閱工作流程 Studio 的 Config 模式。

步驟 6：運行狀態機

狀態機器執行是執行工作流程以執行工作的執行個體。

1. 在 [狀態機器] 頁面上，選擇HelloWorld狀態機器。

2. 在HelloWorld頁面上，選擇 [開始執行]。
3. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

4. 在「輸入」方塊中，以 JSON 格式輸入執行的輸入值。根據您的輸入，`IsHelloWorldExample`變數決定要執行的狀態機器流程。現在，使用以下輸入值：

   ```
   {
      "IsHelloWorldExample": true
   }
   ```

   > **Note**
   >
   > 雖然指定執行輸入是選擇性的，但在本教學課程中，必須指定類似於上述範例輸入的執行輸入。執行狀態機器時，會在`Choice`狀態中參照此輸入值。

5. 選擇 Start execution (開始執行)。
6. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

   在本自學課程中，如果您輸入的輸入值為`"IsHelloWorldExample": true`，您應該會看到下列輸出：

   ```
   {
      "IsHelloWorldExample": true
   },
   {
      "IsHelloWorldExample": true
   ```

```
}
```

步驟 7：更新狀態機

當您更新狀態機時，您的更新最終會保持一致。在短暫的時間之後，所有新啟動的執行都將反映狀態機的更新定義。所有當前正在運行的執行都將在先前的定義下運行到完成。

在此步驟中，您將在工作流程 Studio 的設計模式模式下更新狀態機器。您將在「通行證」狀態中新增一個名為「世界」的`Result`欄位。

1. 在標題為您的執行 ID 的頁面上，選擇編輯狀態機器。
2. 請確定您處於 [設計] 模式。
3. 在畫布上選擇名為「世界」的「通過」狀態，然後選擇「輸出」。
4. 在「結果」方塊中，輸入**`"World has been updated!"`**。
5. 選擇儲存。
6. (選擇性) 在「定義」區域中，檢視工作流程的更新後 Amazon States 語言定義。

```
{
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Hello",
          "States": {
            "Hello": {
              "Type": "Pass",
              "End": true
            }
          }
        },
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "Result": "World has been updated!",
              "End": true
            }
          }
```

```
        }
      ],
      "Next": "PassState"
    }
```

7. 選擇 Execute (執行)。
8. 在新索引標籤中開啟的 [開始執行] 對話方塊中，提供下列執行輸入。

```
{
   "IsHelloWorldExample": true
}
```

9. 選擇 Start Execution (開始執行)。
10. (選擇性) 在「圖表」檢視中，選擇「世界」步驟，然後選擇「輸出」。輸出是世界已經更新！

步驟 8：清除

若要刪除狀態機

1. 從導覽功能表中，選擇 [狀態機]。
2. 在 [狀態電腦] 頁面上，選取 HelloWorld，然後選擇 [刪除]。
3. 在 [刪除狀態機器] 對話方塊中，輸入**`delete`**以確認刪除。
4. 選擇刪除。

   如果刪除成功，畫面頂端會出現綠色的狀態列。綠色狀態列會告訴您狀態機已標記為要刪除。當所有進行中的執行停止運行時，您的狀態機將被刪除。

若要刪除您的執行角色

1. 開啟 IAM 的「角色」頁面。
2. 選擇為您建立的 Step Functions 的 IAM 角色。例如，StepFunctions-HelloWorld 角色- 例如。
3. 選擇 Delete role (刪除角色)。
4. 選擇 Yes, delete (是，刪除)。

Step Functions 的教學課程

本節中的教學可協助您了解使用 AWS Step Functions 的不同層面。

若要完成這些教學課程，您需要一個 AWS 帳戶。如果您沒有 AWS 帳戶，請瀏覽至 https://aws.amazon.com/ 並選擇「建立 AWS 帳戶」。

主題

建立使用 Lambda 的 Step Functions 狀態機器

在本教學課程中，您將使用 AWS Step Functions 來叫用 AWS Lambda 函數來建立單一步驟工作流程。

> **Note**
>
> Step Functions 以狀態機器和任務為基礎。在 Step Functions 中，工作流程稱為狀態機器，該狀態機器是一系列事件驅動的步驟。工作流程中的每個步驟稱為狀態。任務狀態代表另一個 AWS 服務 (例如) 執行的工作單位。 AWS Lambda任務狀態可以調用任何 AWS 服務 或 API。

如需詳細資訊，請參閱：

- 什麼是 AWS Step Functions？
- 致電其他 AWS 服務

Lambda 非常適合Task各州，因為 Lambda 函數是無伺服器且容易撰寫的。您可以在 AWS Management Console 或您最喜歡的編輯器中編寫代碼。 AWS 處理為您的功能和運行它提供計算環境的詳細信息。

在本主題中：

步驟 1：建立 Lambda 函數

您的 Lambda 函數會接收事件資料並傳回問候訊息。

> **Important**
>
> 確保您的 Lambda 函數與狀態機位於相同的 AWS 帳戶和 AWS 區域。

1. 開啟 Lambda 主控台，然後選擇建立函數。
2. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。
3. 針對 函數名稱 ，請輸入 `HelloFunction`。
4. 保留所有其他選項的預設選項，然後選擇 [建立函數]。
5. 建立 Lambda 函數後，複製頁面右上角顯示的函數的 Amazon 資源名稱 (ARN)。若要複製 ARN，請按一
   下
   以下是 ARN 的示例：

   ```
   arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
   ```

6. 將 Lambda 函數的下列程式碼複製到***HelloFunction***頁面的程式碼來源區段中。

```
export const handler = async(event, context, callback) => {
    callback(null, "Hello from " + event.who + "!");
};
```

此程式碼會使用輸入資料的 who 欄位來組合問候語，該欄位由傳送到您函數的 event 物件提供。稍後您會在開始新的執行時，為此函數新增輸入資料。callback 方法會傳回函數中的組合問候語。

7. 選擇部署。

步驟 2：測試 Lambda 函數

測試您的 Lambda 函數以查看它在運作中。

1. 選擇 測試。
2. 事件名稱輸入 HelloEvent。
3. 將事件 JSON 資料取代為下列項目。

```
{
    "who": "AWS Step Functions"
}
```

"who"項目對應於 Lambda 函數中的event.who欄位，完成問候語。運行狀態機時，您將輸入相同的輸入數據。

4. 選擇儲存，然後選擇測試。
5. 若要檢閱測試結果，在 Execution result (執行結果) 下，展開 Details (詳細資訊)。

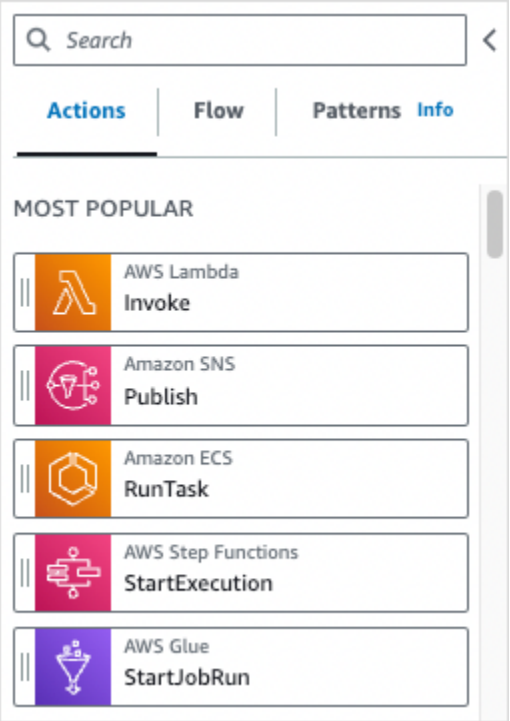
步驟 3：建立狀態機

使用 Step Functions 數主控台建立狀態機器，以呼叫您在步驟 1 中建立的 Lambda 函數。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。

> **Important**
>
> 請確定您的狀態機與先前建立的 Lambda 函數位於相同的 AWS 帳戶和區域。

2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 在左側的「狀態」瀏覽器中，確定您已選取「動作」索引標籤。然後，執行下列動作：
   - 將AWS Lambda 調用 API 拖放到標記為「拖動第一個狀態」的空白狀態。



5. 在右側的「Inspector」面板中，設定 Lambda 函數：
   a. 在「API 參數」區段中，選擇您先前在「函數名稱」下拉式清單中建立的 Lambda 函數。
   b. 將預設選取項保留在裝載下拉式清單中。
6. (選擇性) 選擇「定義」以檢視狀態機器 Amazon States Language (ASL) 定義，該定義會根據您在「動作」標籤和「Inspector 查程式」面板中的選擇自動產生。
7. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   例如，輸入名稱 **`LambdaStateMachine`**。

> **Note**
>
> 狀態機器、執行項目和活動工作的名稱長度不得超過 80 個字元。這些名稱對於您的帳戶和 AWS 地區而言必須是唯一的，且不得包含以下任何一項：
>
> - 空白符號
> - 萬用字元 (? *)
> - 括號字元 (< > { } [ ])
> - 特殊字元 (: ; , \ | ^ ~ $ # % & ` ")
> - 控制字符 (\\u0000-\\u001f 或 \\u007f-\\u009f) 。
>
> 如果您的狀態機器的類型為 Express，則可以為狀態機器的多個執行提供相同的名稱。即使多個執行具有相同的名稱，Step Functions 也會為每個 Express 狀態機器執行產生唯一的執行 ARN。
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

8. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器設定中的所有預設選項。

9. 選擇建立。

10. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

    您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 4：運行狀態機

建立狀態機之後，您可以執行它。

1. 在 [狀態電腦] 頁面上，選擇LambdaStateMachine。
2. 選擇 Start execution (開始執行)。

   此時會顯示「開始執行」對話方塊。
3. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

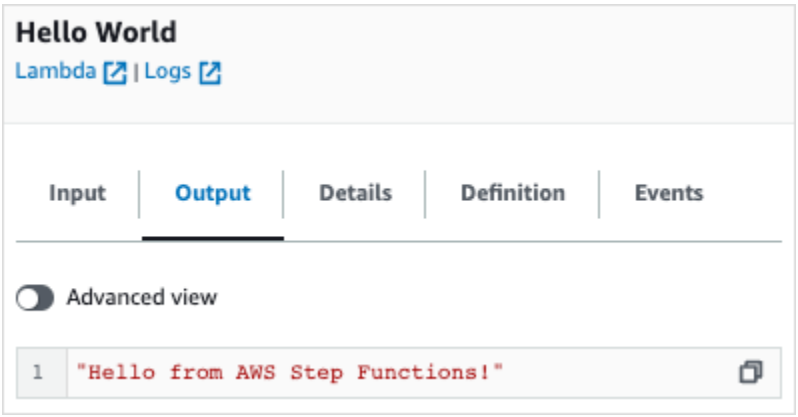
   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。
4. 在「輸入」區域中，以下列項目取代範例執行資料。

   ```
   {
       "who" : "AWS Step Functions"
   }
   ```

   "who"是您的 Lambda 函數用來獲取問候人員姓名的密鑰名稱。
5. 選擇 Start Execution (開始執行)。

   您的狀態機的執行開始，並顯示一個顯示正在運行執行的新頁面。
6. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。



> **Note**
>
> 您也可以在從狀態機器叫用 Lambda 時傳遞承載。如需透過在`Parameters`欄位中傳遞承載來叫用 Lambda 的詳細資訊和範例，請參閱使用 Step Functions 叫用 Lambda。

使用 Step Functions 狀態機處理錯誤條件

在此自學課程中，您將建立具有后援国家欄位的 AWS Step Functions 狀態機器。此`Catch`欄位會根據錯誤訊息類型使用條件式邏輯來回應 AWS Lambda 函數。此技術稱為函數錯誤處理。

如需詳細資訊，請參閱AWS Lambda 開發人員指南中的 Node.js 中的AWS Lambda 函式錯誤。

> **Note**
>
> 您也可以建立在逾時時重試的狀態機器，或是在發生錯誤或逾時時用來轉換`Catch`至特定狀態的狀態機器。如需錯誤處理技巧的範例，請參閱使用重試和擷取的範例。

在本主題中：

- 步驟 1：建立失敗的 Lambda 函數
- 步驟 2：測試 Lambda 函數
- 步驟 3：使用 Catch 欄位建立狀態機
- 步驟 4：運行狀態機

步驟 1：建立失敗的 Lambda 函數

使用 Lambda 函數來模擬錯誤狀況。

> **Important**
>
> 確保您的 Lambda 函數與狀態機位於相同的 AWS 帳戶和 AWS 區域。

1. 開啟主 AWS Lambda 控台，網址為 https://console.aws.amazon.com/lambda/。
2. 選擇建立函數。
3. 選擇 [使用藍圖]，在搜尋方塊`step-functions`中輸入，然後選擇 [擲回自訂錯誤藍圖]。
4. 針對 函數名稱 ，請輸入 `FailFunction`。
5. 對於角色，請保留預設選項 (使用基本 Lambda 權限建立新角色)。
6. 下列程式碼會顯示在 Lambda 函數程式碼窗格中。

```
exports.handler = async (event, context) => {
    function CustomError(message) {
        this.name = 'CustomError';
        this.message = message;
    }
    CustomError.prototype = new Error();

    throw new CustomError('This is a custom error!');
};
```

   `context` 物件會傳回錯誤訊息 `This is a custom error!`。
7. 選擇建立函數。
8. 建立 Lambda 函數後，複製頁面右上角顯示的函數的 Amazon 資源名稱 (ARN)。若要複製 ARN，請按一下。
   以下是 ARN 的示例：

```
arn:aws:lambda:us-east-1:123456789012:function:FailFunction
```

9. 選擇部署。

步驟 2：測試 Lambda 函數

測試您的 Lambda 函數以查看它在運作中。

1. 在FailFunction頁面上，選擇 [測試] 索引標籤，然後選擇 [測試]。您不需要建立測試事件。
2. 若要檢閱測試結果 (模擬錯誤)，請在執行結果下展開詳細資料。

步驟 3：使用 Catch 欄位建立狀態機

使用 Step Functions 主控台建立使用具有Catch欄位之任務狀態的狀態機器。在「工作」狀態下新增對 Lambda 函數的參考。狀態機器調用 Lambda 函數，該函數在執行期間失敗。Step Functions 會在重試之間使用指數輪詢來重試函式兩次。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 選擇 [程式碼] 以開啟程式碼編輯器。在程式碼編輯器中，您可以撰寫並編輯工作流程的 Amazon States Language (ASL) 定義。
5. 貼上下列程式碼，但取代您先前在Resource欄位中建立的 Lambda 函數的 ARN。

```
{
   "Comment": "A Catch example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "CreateAccount",
   "States": {
      "CreateAccount": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["CustomError"],
            "Next": "CustomErrorFallback"
         }, {
            "ErrorEquals": ["States.TaskFailed"],
            "Next": "ReservedTypeFallback"
         }, {
            "ErrorEquals": ["States.ALL"],
            "Next": "CatchAllFallback"
         } ],
         "End": true
```

```
      },
      "CustomErrorFallback": {
         "Type": "Pass",
         "Result": "This is a fallback from a custom Lambda function exception",
         "End": true
      },
      "ReservedTypeFallback": {
         "Type": "Pass",
         "Result": "This is a fallback from a reserved error code",
         "End": true
      },
      "CatchAllFallback": {
         "Type": "Pass",
         "Result": "This is a fallback from any error code",
         "End": true
      }
   }
}
```

這是您使用 Amazon 州語言的狀態機器的描述。它定義名為 `CreateAccount` 的單一 `Task` 狀態。如需詳細資訊，請參閱狀態機器結構。

如需 `Retry` 欄位語法的詳細資訊，請參閱使用重試和使用 Catch 的狀態機器示例。

> **Note**
>
> Lambda 中未處理的錯誤會報告為錯誤輸出`Lambda.Unknown`中。這些包括 out-of-memory 錯誤和功能超時。您可以在`Lambda.UnknownStates.ALL`、或上進行比對`States.TaskFailed`來處理這些錯誤。當 Lambda 達到調用的最大數量時，錯誤為。`Lambda.TooManyRequestsException`如需 Lambda 函數錯誤的詳細資訊，請參閱AWS Lambda 開發人員指南中的錯誤處理和自動重試。

6. (選擇性) 在中圖形視覺化窗格，查看工作流程的即時圖形視覺效果。
7. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學，輸入 **`Catchfailure`**。
8. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器設定中的所有預設選項。

9. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

   > **Note**
   >
   > 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 4：運行狀態機

建立狀態機之後，您可以執行它。

1. 在 [狀態電腦] 頁面上，選擇 [捕捉失敗]。
2. 在「捕捉失敗」頁面上，選擇「開始執行」。此時會顯示「開始執行」對話方塊。
3. 在 [開始執行] 對話方塊中，執行下列動作：

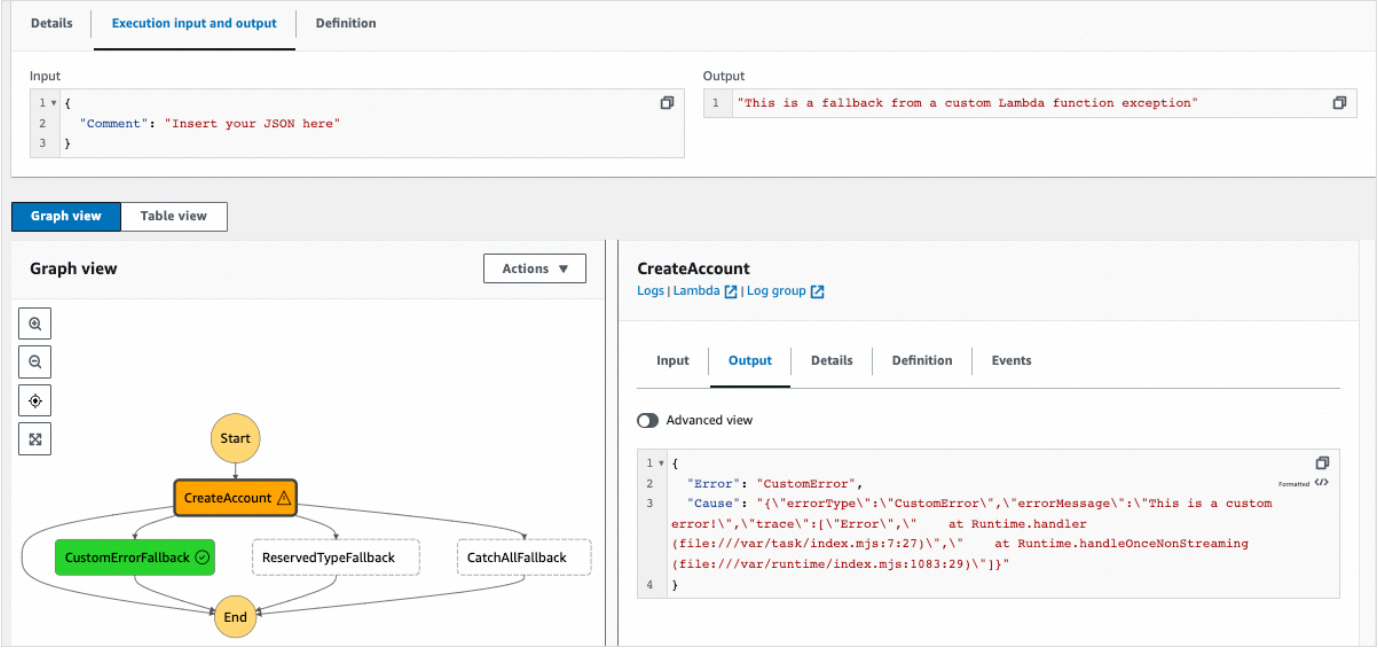
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。
   3. 選擇 Start execution (開始執行)。
   4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

例如，若要檢視您的自訂錯誤訊息，請在「圖表」檢視中選擇CreateAccount步驟，然後選擇「輸出」標籤。



> **Note**
>
> 您可以使用 ResultPath 來保留狀態輸入與錯誤。請參閱用於 ResultPath 將錯誤和輸入都包含在 Catch。

使用內嵌對應狀態重複動作

本教程可幫助您開始在內聯模式下使用Map狀態。您可以在工作流程中使用內嵌對應狀態來重複執行動作。如需有關內嵌模式的詳細資訊，請參閱在內嵌模式中對應狀態。

在本教學課程中，您會使用內嵌對應狀態來重複產生第 4 版的通用唯一識別碼 (v4 UUID)。您可以先建立包含兩個Pass狀態和工作流程 Studio 中的內嵌對應狀態的工作流程。然後，您可以設定輸入和輸出，包括Map狀態的輸入 JSON 陣列。該Map狀態返回一個輸出數組，該數組包含為輸入數組中的每個項目生成的 v4 UUID。

目錄

- 步驟 3：檢閱自動產生的 Amazon 州語言定義並儲存工作流程
- 步驟 4：運行狀態機

步驟 1：建立工作流程原型

在此步驟中，您可以使用工作流程 Studio 建立工作流程的原型。工作流程 Studio 是 Step Functions 控制台中提供的可視化工作流程設計器。您將從 [流程] 索引標籤中選擇所需的狀態，並使用工作流程 Studio 的拖放功能來建立工作流程原型。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 從「流程」索引標籤中，將「暫不處理」狀態拖放至標示為「拖曳第一個狀態」的空白狀態。
5. 將「地圖」狀態拖曳至「通過」狀態下方。將 [對映] 狀態重新命名為**Map demo**。
6. 拖曳第二個傳遞狀態並將其放入地圖示範狀態內。
7. 將第二個「通過」狀態重新命名為 **Generate UUID**

步驟 2：配置輸入和輸出

在此步驟中，您可以為工作流程原型中的所有狀態配置輸入和輸出。首先，您可以使用第一個「通過」狀態將某些固定資料注入工作流程。此通過狀態傳遞這個數據作為輸入到地圖演示狀態。在此輸入中，您可以指定包含地圖演示狀態應迭代輸入數組的節點。然後，您可以定義地圖演示狀態應重複以生成 v4 UUID 的步驟。最後，您可以將輸出設定為每次重複傳回。

1. 選擇工作流程原型中的第一個「通過」狀態。在「輸出」頁籤的「結果」下輸入下列內容：

```
{
  "foo": "bar",
  "colors": [
    "red",
    "green",
    "blue",
    "yellow",
    "white"
  ]
}
```

2. 選擇 [對應] 示範狀態，然後在 [設定] 索引標籤中執行下列動作：

   a. 選擇 [提供項目陣列的路徑]。

   b. 指定下列參照路徑，以選取包含輸入陣列的節點：

   ```
   $.colors
   ```

3. 選擇「產生 UUID」狀態，然後在「輸入」索引標籤中執行下列動作：

   a. 選擇使用參數轉換輸入。

   b. 輸入下列 JSON 輸入，為每個輸入陣列項目產生 v4 UUID。您可以使用`States.UUID`內建函數來產生 UUID。

   ```
   {
     "uuid.$": "States.UUID()"
   }
   ```

4. 針對「產生 UUID」狀態，選擇「輸出」索引標籤，然後執行下列動作：

   a. 選擇篩選輸出方式 OutputPath。

   b. 輸入下列參考路徑，以選取包含輸出陣列項目的 JSON 節點：

   ```
   $.uuid
   ```

步驟 3：檢閱自動產生的 Amazon 州語言定義並儲存工作流程

當您將狀態從「流程」面板拖放到畫布上時，工作流程 Studio 會自動即時編寫工作流程的 Amazon 狀態語言 (ASL) 定義。您可以視需要編輯此定義。

1. (選擇性) 選擇Inspector面板上的「定義」，以檢視自動產生的工作流程 Amazon States 語言定義。

   > **Tip**
   >
   > 您也可以在工作流程工作室中檢視 ASL 定義。程式碼編輯器在程式碼編輯器中，您也可以編輯工作流程的 ASL 定義。

下列範例顯示為您的工作流程自動產生的 Amazon 狀態語言定義。

```
{
    "Comment": "Using Map state in Inline mode",
    "StartAt": "Pass",
    "States": {
      "Pass": {
        "Type": "Pass",
        "Next": "Map demo",
        "Result": {
          "foo": "bar",
          "colors": [
            "red",
            "green",
            "blue",
            "yellow",
            "white"
          ]
        }
      },
      "Map demo": {
        "Type": "Map",
        "ItemsPath": "$.colors",
        "ItemProcessor": {
          "ProcessorConfig": {
            "Mode": "INLINE"
          },
          "StartAt": "Generate UUID",
          "States": {
            "Generate UUID": {
              "Type": "Pass",
              "End": true,
              "Parameters": {
                "uuid.$": "States.UUID()"
              },
              "OutputPath": "$.uuid"
            }
          }
        },
        "End": true
      }
    }
```

```
}
```

2. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **InlineMapDemo**。

3. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器組態中的所有預設選項。

4. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 4：運行狀態機

狀態機器執行是執行工作流程以執行工作的執行個體。

1. 在InlineMapDemo頁面上，選擇 [開始執行]。
2. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。
   3. 選擇 Start execution (開始執行)。

4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

若要檢視執行輸入和輸出 side-by-side，請選擇執行輸入和輸出。在「輸出」下，檢視由Map狀態傳回的輸出陣列。下面是輸出數組的一個例子：

```
[
  "a85cbc7b-4e65-4ac2-97af-80ed504adc1d",
  "b05bca11-d481-414e-aa9a-88285ec6590d",
  "f42d59f7-bd32-480f-b270-caddb518ce2a",
  "15f18616-517d-4b69-b7c3-bf22222d2efd",
  "690bcfee-6d58-408c-a6b4-1995ccafdbd2"
]
```

使用分散式地圖複製大規模 CSV 資料

本教學課程可協助您開始在分散式模式下使用Map狀態。設置為分佈式的Map狀態稱為分佈式地圖狀態。您可以在工作流程中使用分散式地圖狀態來迭代大規模的 Amazon S3 資料來源。狀Map態會將每個版序當做子工作流程執行來執行，以達到高並行性。如需分散式模式的詳細資訊，請參閱分散式模式中的對應狀態。

在本教學中，您會使用分散式地圖狀態在 Amazon S3 儲存貯體中迭代 CSV 檔案。然後，您將其內容與子工作流程執行的 ARN 一起傳回到另一個 Amazon S3 儲存貯體。您可以先在工作流程 Studio 中建立工作流程原型。接下來，您將Map狀態的處理模式設定為 [分散式]、將 CSV 檔案指定為資料集，並將其位置提供給Map狀態。您也可以指定子工作流程執行的工作流程類型，分散式對應狀態開始為 Express。

除了這些設定之外，您還可以針對本教學課程中使用的範例工作流程指定其他組態，例如同時執行子項工作流程的最大數目，以及匯出Map結果的位置。

目錄

- 第 2 步：配置地圖狀態的必填字段
- 步驟 3：設定其他選項
- 步驟 4：設定 Lambda 函數
- 步驟 5：更新工作流程原型
- 步驟 6：檢閱自動產生的 Amazon 州語言定義並儲存工作流程
- 步驟 7：運行狀態機

必要條件

- 將 CSV 檔案上傳到 Amazon S3 儲存貯體。您必須在 CSV 檔案中定義標題列。如需有關 CSV 檔案的大小限制以及如何指定標頭列的資訊，請參閱Amazon S3 儲存貯體中的 CSV 檔案。
- 建立另一個 Amazon S3 儲存貯體和該儲存貯體內的資料夾，以將狀Map態結果匯出到。

> **Important**
>
> 確保您的 Amazon S3 儲存貯體 AWS 帳戶 與狀態機器位於相 AWS 區域 同的狀態機器下。

步驟 1：建立工作流程原型

在此步驟中，您可以使用工作流程 Studio 建立工作流程的原型。工作流程 Studio 是 Step Functions 控制台中提供的可視化工作流程設計器。您可以分別從「流程」和「動作」 標籤中選擇所需的狀態和 API 動作。您將使用拖放工作流 Studio 的功能來創建工作流原型。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 從「流程」索引標籤中，將地圖狀態拖放至標示為「在此處拖曳第一個狀態」的空白狀態。
5. 在組態索引標籤中，對於狀態名稱，輸入**`Process data`**。
6. 從「動作」索引標籤中，將「AWS Lambda 叫用 API」動作拖放到「處理程序」資料狀態中。
7. 將AWS Lambda 呼叫狀態重新命名為**`Process CSV data`**。

第 2 步：配置地圖狀態的必填字段

在此步驟中，您可以設定「分散式對應」狀態的下列必要欄位：

- ItemReader— 指定資料集及其Map狀態可從中讀取輸入的位置。
- ItemProcessor— 指定下列值：
  - ProcessorConfig— ExecutionType 將Mode和設定為DISTRIBUTED和EXPRESS分別。這會設定Map狀態的處理模式和「分散式對映」狀態啟動之子工作流程執行的工作流程類型。
  - StartAt—「地圖」工作流程中的第一個狀態。
  - States— 定義 Map 工作流程，這是每個子工作流程執行中要重複的一組步驟。
- ResultWriter— 指定 Step Functions 寫入分散式地圖狀態結果的 Amazon S3 位置。

> **Important**
>
> 確保用於匯出 Map Run 結果的 Amazon S3 儲存貯體與您的狀態機器處於相 AWS 區域 同 AWS 帳戶 的狀態機器下。否則，您的狀態機執行將失敗並顯示States.ResultWriterFailed錯誤。

若要設定必要欄位：

1. 選擇處理資料狀態，然後在組態索引標籤中執行下列動作：
   a. 對於「處理」模式，請選擇「分散
   b. 對於項目來源，請選擇 Amazon S3，然後從 S3 項目來源下拉式清單中選擇 S3 中的 CSV 檔案。
   c. 執行下列動作以指定 CSV 檔案的 Amazon S3 位置：
      i. 對於 S3 物件，請從下拉式清單中選取 [輸入儲存貯體和金鑰]。
      ii. 在儲存貯體中，輸入包含 CSV 檔案的 Amazon S3 儲存貯體的名稱。例如 **sourceBucket**。
      iii. 在金鑰中，輸入您儲存 CSV 檔案的 Amazon S3 物件名稱。您也必須在此欄位中指定 CSV 檔案的名稱。例如 **csvDataset/ratings.csv**。
   d. 對於 CSV 檔案，您還必須指定欄標題的位置。若要這麼做，請選擇 [其他組態]，然後針對 CSV 標頭位置保留預設選取 [第一列] (如果 CSV 檔案的第一列是標頭)。否則，請選擇「給定」以在狀態機定義中指定標頭。如需詳細資訊，請參閱 ReaderConfig。

e. 針對子項執行類型，選擇快速。

2. 在匯出位置中，若要將地圖執行結果匯出到特定的 Amazon S3 位置，請選擇將地圖狀態的輸出匯出到 Amazon S3。

3. 請執行下列操作：

   a. 對於 S3 儲存貯體，請從下拉式清單中選擇 [輸入儲存貯體名稱和前綴]

   b. 在儲存貯體中，輸入您要將結果匯出到的 Amazon S3 儲存貯體的名稱。例如 **mapOutputs**。

   c. 在「字首」中，輸入您要儲存結果的資料夾名稱。例如 **resultData**。

步驟 3：設定其他選項

除了分散式貼圖狀態的必要設定之外，您還可以指定其他選項。這些可能包括同時執行子項工作流程的最大數目，以及將Map狀態結果匯出至的位置。

1. 選擇處理資料狀態。然後，在項目來源中，選擇其他組態。

2. 請執行下列操作：

   a. 選擇 [修改項目]，為每個子工作流程執行指定自訂 JSON 輸入。 ItemSelector

   b. 輸入下列 JSON 輸入：

```
{
  "index.$": "$$.Map.Item.Index",
  "value.$": "$$.Map.Item.Value"
}
```

   若要取得有關如何建立自訂輸入的資訊，請參閱〈〉ItemSelector。

3. 在「執行階段」設定中，對於並行限制，指定「分散式對應」狀態可以啟動的並行子工作流程執行數目。例如，輸入 **100**。

4. 在瀏覽器上開啟新視窗或索引標籤，並完成您將在此工作流程中使用的 Lambda 函數的設定，如中所述步驟 4：設定 Lambda 函數。

步驟 4：設定 Lambda 函數

> **Important**
> 確保您的 Lambda 函數與狀態機 AWS 區域 相同。

1. 開啟 Lambda 主控台，然後選擇建立函數。
2. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。
3. 在「基本資訊」區段中，設定 Lambda 函數：
   a. 針對 函數名稱 ，請輸入 **distributedMapLambda**。
   b. 針對 執行時間，請選擇 Node.js 16.x。
   c. 保留所有默認選擇並選擇創建功能。
   d. 建立 Lambda 函數後，複製頁面右上角顯示的函數的 Amazon 資源名稱 (ARN)。您需要在工作流程原型中提供此功能。若要複製 ARN，請按一下
   以下是 ARN 的示例：

```
arn:aws:lambda:us-east-2:123456789012:function:distributedMapLambda
```

4. 複製 Lambda 函數的下列程式碼，並將其貼到distributedMapLambda頁面的程式碼來源區段中。

```
exports.handler = async function(event, context) {
    console.log("Received Input:\n", event);

    return {
        'statusCode' : 200,
        'inputReceived' : event //returns the input that it received
    }
};
```

5. 選擇部署。部署函數後，請選擇「測試」以查看 Lambda 函數的輸出。

步驟 5：更新工作流程原型

在 Step Functions 數主控台中，您將更新工作流程以新增 Lambda 函數的 ARN。

1. 返回建立工作流程原型的標籤或視窗。
2. 選擇「處理 CSV 資料」步驟，然後在「組態」索引標籤中執行下列操作：
   a. 針對整合類型，選擇最佳化。
   b. 在函數名稱中，開始輸入 Lambda 函數的名稱。從出現的下拉式清單中選擇函數，或選擇 [輸入函數名稱] 並提供 Lambda 函數 ARN。

步驟 6：檢閱自動產生的 Amazon 州語言定義並儲存工作流程

當您將狀態從「動作」和「流程」索引標籤拖放到畫布上時，Work flow Studio 會自動即時編寫工作流程的 Amazon States 語言定義。您可以視需要編輯此定義。

1. (選擇性) 在Inspector面板上選擇「定義」並檢視狀態機定義。

> **Tip**
>
> 您也可以在工作流程工作室中檢視 ASL 定義。程式碼編輯器在程式碼編輯器中，您也可以編輯工作流程的 ASL 定義。

下列範例程式碼顯示為您的工作流程自動產生的 Amazon States 語言定義。

```
{
  "Comment": "Using Map state in Distributed mode",
  "StartAt": "Process data",
  "States": {
    "Process data": {
      "Type": "Map",
      "MaxConcurrency": 100,
      "ItemReader": {
        "ReaderConfig": {
          "InputType": "CSV",
          "CSVHeaderLocation": "FIRST_ROW"
        },
        "Resource": "arn:aws:states:::s3:getObject",
        "Parameters": {
          "Bucket": "sourceBucket",
          "Key": "csvDataset/ratings.csv"
        }
      },
```

```
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "EXPRESS"
        },
        "StartAt": "Process CSV data",
        "States": {
          "Process CSV data": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
              "Payload.$": "$",
              "FunctionName": "arn:aws:lambda:us-
east-2:123456789012:function:distributedMapLambda"
            },
            "End": true
          }
        }
      },
      "Label": "Processdata",
      "End": true,
      "ResultWriter": {
        "Resource": "arn:aws:states:::s3:putObject",
        "Parameters": {
          "Bucket": "mapOutputs",
          "Prefix": "resultData"
        }
      },
      "ItemSelector": {
        "index.$": "$$.Map.Item.Index",
        "value.$": "$$.Map.Item.Value"
      }
    }
  }
}
```

2. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **DistributedMapDemo**。

3. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

在本教學課程中，請保留狀態機器組態中的所有預設選項。

4. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 7：運行狀態機

執行是您執行工作流程以執行工作的狀態機器執行個體。

1. 在DistributedMapDemo頁面上，選擇 [開始執行]。
2. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。
   3. 選擇 Start execution (開始執行)。
   4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

例如，選擇Map狀態，然後選擇 [對應執行] 以開啟 [對映執行詳細資訊] 頁面。在此頁面上，您可以檢視分散式對應狀態的所有執行詳細資訊，以及其啟動的子工作流程執行項目。如需有關此頁面的資訊，請參閱檢查地圖運行。

使用 Lambda 函數處理整批資料

在本教學課程中，您會使用「分散式地圖」狀態的ItemBatcher欄位來處理 Lambda 函數內的整批項目。每個批次最多包含三個項目。「分散式對應」狀態會啟動四個子工作流程執行，其中每個執行都會處理三個項目，而一個執行則會處理單一項目。每個子工作流程執行都會叫用 Lambda 函數，該函數會重複執行批次中存在的個別項目。

您將創建一個狀態機，該狀態機器對整數數組執行乘法。假設您提供的整數數組作為輸入是`[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`，乘法因子為7。然後，將這些整數乘以 `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]` 7 的因子之後形成的結果數組。

主題

- 步驟 1：建立狀態機
- 步驟 2：建立 Lambda 函數
- 步驟 3：運行狀態機

步驟 1：建立狀態機

在此步驟中，您會建立狀態機器的工作流程原型，將整批資料傳遞至您將在步驟 2 中建立的 Lambda 函數。

- 使用下列定義可使用 Step Functions 主控台建立狀態機器。如需有關建立狀態機器的資訊，請參閱〈使用分散式地圖狀態入門〉自學課程步驟 1：建立工作流程原型中的〈〉。

  在這個狀態機中，您可以定義一個分散式地圖狀態，該狀態接受 10 個整數作為輸入的陣列，並將此陣列分批傳遞給 Lambda 函數3。Lambda 函數迭代存在於批處理中的各個項目，並返回一個名為`multiplied`的輸出數組。輸出陣列包含對輸入陣列中傳遞的項目執行乘法的結果。

> **Important**
>
> 請務必將下列程式碼中的 Lambda 函數的 Amazon 資源名稱 (ARN) 取代為您將在步驟 2 中建立之函數的 ARN。

```
{
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "Map",
      "Result": {
        "MyMultiplicationFactor": 7,
        "MyItems": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      }
    },
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "Lambda Invoke",
        "States": {
          "Lambda Invoke": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
              "Payload.$": "$",
              "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:functionName"
            },
            "Retry": [
              {
                "ErrorEquals": [
                  "Lambda.ServiceException",
                  "Lambda.AWSLambdaException",
                  "Lambda.SdkClientException",
```

```
                  "Lambda.TooManyRequestsException"
                ],
                "IntervalSeconds": 2,
                "MaxAttempts": 6,
                "BackoffRate": 2
              }
            ],
            "End": true
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems"
    }
  }
}
```

步驟 2：建立 Lambda 函數

在此步驟中，您會建立 Lambda 函數來處理批次中傳遞的所有項目。

> **Important**
>
> 確保您的 Lambda 函數與狀態機 AWS 區域 相同。

建立 Lambda 函數

1. 使用 L ambda 主控台建立名為的 Python 3.9 Lambda 函數**ProcessEntireBatch**。如需建立 Lambda 函數的相關資訊，請參閱〈使用分散式地圖狀態入門〉教學課程中的步驟 4：設定 Lambda 函數。
2. 複製 Lambda 函數的下列程式碼，並將其貼到 Lambda 函數的程式碼原始碼區段中。

```
import json

def lambda_handler(event, context):
    multiplication_factor = event['BatchInput']['MyMultiplicationFactor']
    items = event['Items']

    results = [multiplication_factor * item for item in items]

    return {
        'statusCode': 200,
        'multiplied': results
    }
```

3. 建立 Lambda 函數之後，請複製頁面右上角顯示的函數 ARN。若要複製 ARN，請按一下。 以下是 ARN 範例，其中*function-name*是 Lambda 函數的名稱 (在本例中為ProcessEntireBatch)：

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

   您需要在步驟 1 中創建的狀態機中提供函數 ARN。

4. 選擇部署以部署變更。

步驟 3：運行狀態機

當您執行狀態機器時，分散式地圖狀態會啟動四個子工作流程執行，其中每個執行會處理三個項目，而一個執行則處理單一項目。

下列範例顯示其中一個子工作流程執行傳遞至ProcessEntireBatch函數的資料。

```
{
  "BatchInput": {
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

根據此輸入，下列範例會顯示由 Lambda 函數傳回multiplied的名為的輸出陣列。

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

狀態機器會傳回下列輸出，其中包含四個以四個子工作流程執行命名multiplied的陣列。這些陣列包含個別輸入項目的乘法結果。

```
[
  {
    "statusCode": 200,
    "multiplied": [7, 14, 21]
  },
  {
    "statusCode": 200,
    "multiplied": [28, 35, 42]
  },
  {
    "statusCode": 200,
    "multiplied": [49, 56, 63]
  },
  {
    "statusCode": 200,
    "multiplied": [70]
  }
]
```

要將返回的所有數組項合併到單個輸出數組中，可以使用該ResultSelector字段。在「分佈式地圖」狀態中定義此字段以查找所有數multiplied組，提取這些數組中的所有項目，然後將它們合併為單個輸出數組。

若要使用ResultSelector欄位，請更新狀態機定義，如下列範例所示。

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
```

```
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

更新後的狀態機會傳回合併的輸出陣列，如下列範例所示。

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

使用 Lambda 函數處理個別資料項目

在本教學課程中，您可以使用「分散式地圖」狀態的ItemBatcher欄位，使用 Lambda 函數對批次中存在的個別項目進行迭代。「分散式對應」狀態會啟動四個子工作流程執行。這些子工作流程中的每一個都會執行內嵌對映狀態。對於其每次迭代，內聯映射狀態調用 Lambda 函數，並將單個項目從批處理傳遞給函數。Lambda 函數接著會處理項目並傳回結果。

您將創建一個狀態機，該狀態機器對整數數組執行乘法。假設您提供的整數數組作為輸入是`[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`，乘法因子為7。然後，將這些整數乘以 `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]` 7 的因子之後形成的結果數組。

主題

- 步驟 1：建立狀態機
- 步驟 2：建立 Lambda 函數
- 步驟 3：運行狀態機

步驟 1：建立狀態機

在此步驟中，您會建立狀態機器的工作流程原型，將單一項目從批次項目傳遞至您將在步驟 2 中建立的 Lambda 函數的每次叫用。

- 使用下列定義可使用 Step Functions 主控台建立狀態機器。如需有關建立狀態機器的資訊，請參閱〈使用分散式地圖狀態入門〉自學課程步驟 1：建立工作流程原型中的〈〉。

在此狀態機器中，您可以定義接受 10 個整數的陣列作為輸入的分散式地圖狀態，並將這些陣列項目批次傳遞給子工作流程執行。每個子工作流程執行都會接收三個項目的批次作為輸入，並執行內嵌對應狀態。內聯地圖狀態的每一次迭代調用 Lambda 函數，並將項目從批處理傳遞給函數。然後，此函數會將項目乘以係數，7並傳回結果。

每個子工作流程執行的輸出都是 JSON 陣列，其中包含每個傳遞項目的乘法結果。

> **Important**
>
> 請務必將下列程式碼中的 Lambda 函數的 Amazon 資源名稱 (ARN) 取代為您將在步驟 2 中建立之函數的 ARN。

```
{
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "Map",
      "Result": {
        "MyMultiplicationFactor": 7,
        "MyItems": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      }
    },
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "InnerMap",
        "States": {
          "InnerMap": {
            "Type": "Map",
            "ItemProcessor": {
              "ProcessorConfig": {
                "Mode": "INLINE"
              },
              "StartAt": "Lambda Invoke",
              "States": {
```

```
                "Lambda Invoke": {
                  "Type": "Task",
                  "Resource": "arn:aws:states:::lambda:invoke",
                  "OutputPath": "$.Payload",
                  "Parameters": {
                    "Payload.$": "$",
                    "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:functionName"
                  },
                  "Retry": [
                    {
                      "ErrorEquals": [
                        "Lambda.ServiceException",
                        "Lambda.AWSLambdaException",
                        "Lambda.SdkClientException",
                        "Lambda.TooManyRequestsException"
                      ],
                      "IntervalSeconds": 2,
                      "MaxAttempts": 6,
                      "BackoffRate": 2
                    }
                  ],
                  "End": true
                }
              }
            },
            "End": true,
            "ItemsPath": "$.Items",
            "ItemSelector": {
              "MyMultiplicationFactor.$": "$.BatchInput.MyMultiplicationFactor",
              "MyItem.$": "$$.Map.Item.Value"
            }
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemsPath": "$.MyItems",
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
```

```
      }
    }
  }
}
```

步驟 2：建立 Lambda 函數

在此步驟中，您會建立 Lambda 函數來處理從批次傳遞的每個項目。

> **Important**
>
> 確保您的 Lambda 函數與狀態機 AWS 區域 相同。

建立 Lambda 函數

1. 使用 L ambda 主控台建立名為的 Python 3.9 Lambda 函數**ProcessSingleItem**。如需建立 Lambda 函數的相關資訊，請參閱〈使用分散式地圖狀態入門〉教學課程中的步驟 4：設定 Lambda 函數。
2. 複製 Lambda 函數的下列程式碼，並將其貼到 Lambda 函數的程式碼原始碼區段中。

```
import json

def lambda_handler(event, context):

    multiplication_factor = event['MyMultiplicationFactor']
    item = event['MyItem']

    result = multiplication_factor * item

    return {
        'statusCode': 200,
        'multiplied': result
    }
```

3. 建立 Lambda 函數之後，請複製頁面右上角顯示的函數 ARN。若要複製 ARN，請按一下。 以下是 ARN 範例，其中*function-name*是 Lambda 函數的名稱 (在本例中為ProcessSingleItem)：

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

您需要在步驟 1 中創建的狀態機中提供函數 ARN。

4. 選擇部署以部署變更。

步驟 3：運行狀態機

當您執行狀態機器時，分散式地圖狀態會啟動四個子工作流程執行，其中每個執行會處理三個項目，而一個執行則處理單一項目。

下列範例顯示傳遞至子工作流程執行內其中一個ProcessSingleItem函數叫用的資料。

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

有了這個輸入，下列範例會顯示 Lambda 函數傳回的輸出。

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

下列範例顯示其中一個子工作流程執行的輸出 JSON 陣列。

```
[
  {
    "statusCode": 200,
    "multiplied": 7
  },
  {
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
```

```
  }
]
```

狀態機會傳回下列輸出，其中包含四個子工作流程執行的四個陣列。這些陣列包含個別輸入項目的乘法結果。

最後，狀態機輸出是一個名為的陣列，結合multiplied了針對四個子工作流程執行傳回的所有乘法結果。

```
[
  [
    {
      "statusCode": 200,
      "multiplied": 7
    },
    {
      "statusCode": 200,
      "multiplied": 14
    },
    {
      "statusCode": 200,
      "multiplied": 21
    }
  ],
  [
    {
      "statusCode": 200,
      "multiplied": 28
    },
    {
      "statusCode": 200,
      "multiplied": 35
    },
    {
      "statusCode": 200,
      "multiplied": 42
    }
  ],
  [
    {
      "statusCode": 200,
      "multiplied": 49
    },
```

```
    {
      "statusCode": 200,
      "multiplied": 56
    },
    {
      "statusCode": 200,
      "multiplied": 63
    }
  ],
  [
    {
      "statusCode": 200,
      "multiplied": 70
    }
  ]
]
```

若要將子工作流程執行傳回的所有乘法結果合併為單一輸出陣列，您可以使用ResultSelector欄位。在「分散式地圖」狀態中定義此欄位以尋找所有結果、擷取個別結果，然後將它們合併為一個名為的單一輸出陣列multiplied。

若要使用ResultSelector欄位，請更新狀態機定義，如下列範例所示。

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
```

```
  }
}
```

更新後的狀態機會傳回合併的輸出陣列，如下列範例所示。

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

啟動狀態機器執行以回應 Amazon S3 事件

您可以執行AWS Step Functions狀態機器以回應 Amazon EventBridge 規則。

本教學說明如何將狀態機器設定為 Amazon EventBridge 規則的目標。當檔案新增至 Amazon Simple Storage Service (Amazon S3) 儲存貯體時，此規則將啟動狀態機器執行。

對於實際的應用程式，您可以啟動狀態機器來對新增至儲存貯體的檔案執行操作，例如建立縮圖或針對影像和視訊檔案執行 Amazon Rekognition 分析。

在本教學中，您將檔案上傳到 Amazon S3 儲存貯體以開始執行`Helloworld`狀態機器。然後，您可以檢閱該執行的範例輸入，以識別傳送到的 Amazon S3 事件通知輸入中包含的資訊 EventBridge。

主題

先決條件：建立狀態機器

您必須先建立狀態機器，才能將狀態機器設定為 Amazon EventBridge 目標。

- 若要建立基本狀態機器，請使用建立使用 Lambda 函數的狀態機教學課程。
- 如果您已經有 `Helloworld` 狀態機器，請移至下一個步驟。

步驟 1：在 Amazon S3 中創建一個存儲桶

現在您已經擁有Helloworld狀態機器，您需要建立用於存放檔案的 Amazon S3 儲存貯體。在本教學課程的步驟 3 中，您會設定規則，以便在檔案上傳至此儲存貯體時 EventBridge 觸發狀態機器的執行。

1. 導覽至 Amazon S3 主控台，然後選擇「建立儲存貯體」以建立要在其中存放檔案的儲存貯體，並觸發 Amazon S3 事件規則。
2. 輸入 Bucket name (儲存貯體名稱)，例如 *username*-sfn-tutorial。

> **Note**
>
> Amazon S3 中所有AWS區域中所有現有儲存貯體名稱的儲存貯體名稱必須是唯一的。使用您自有的#####，以確保此名稱獨一無二。您需要在相同的 AWS 區域中建立所有資源。

3. 保留頁面上的所有預設選項，然後選擇 [建立值區]。

步驟 2：啟用 Amazon S3 事件通知 EventBridge

建立 Amazon S3 儲存貯體之後，請將其設定為在 S3 儲存貯體中發生某些事件 (例如檔案上傳) EventBridge 時將事件傳送至。

1. 導覽至 Amazon S3 主控台。
2. 在儲存貯體名稱清單中，選擇要啟用事件的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 向下捲動頁面以檢視「事件通知」區段，然後在 Amazon 子區 EventBridge段中選擇 「編輯」。
5. 在「 EventBridge 針對此值區中的所有事件傳送通知至 Amazon」下方，選擇「開啟」。
6. 選擇 Save Changes (儲存變更)。

> **Note**
>
> 啟用之後 EventBridge，變更大約需要五分鐘才會生效。

步驟 3：創建一個亞馬遜 EventBridge 規則

在您擁有狀態機器並建立 Amazon S3 儲存貯體並將其設定為傳送事件通知至之後 EventBridge，請建立 EventBridge 規則。

> **Note**
>
> 您必須在與 Amazon S3 儲存貯體相同的AWS區域中設定 EventBridge 規則。

若要建立 規則

1. 導覽至 Amazon 主 EventBridge 控台，選擇建立規則。

   > **Tip**
   >
   > 或者，在 EventBridge 主控台的導覽窗格中，選擇「匯流排」下的「規則」，然後選擇「建立規則」。

2. 輸入規則的「名稱」(例如，*S3Step Functions*)，並選擇性地輸入規則的「摘要」。
3. 對於事件匯流排和規則類型，保留預設選項。
4. 選擇下一步。這會開啟 [建置事件模式] 頁面。
5. 向下捲動至「事件模式」區段，然後執行下列動作：

   a. 對於事件來源，請保留事件或 EventBridge 合作夥伴事件的AWS預設選項。

   b. 對於AWS服務，請選擇 S imple Storage Service (S3)。

   c. 對於事件類型，請選擇 Amazon S3 事件通知。

   d. 選擇 [特定事件]，然後選擇 [建立的物件]。

   e. 依名稱選擇特定值區，然後輸入您在步驟 1 (*username*-sfn-tutorial) 中建立的值區名稱以儲存檔案。

   f. 選擇下一步。這會開啟 [選取目標] 頁面。

建立目標

1. 在目標 1 中，保留預設的AWS服務選項。
2. 在 [選取目標] 下拉式清單中，選取 [Step Functions 狀態機器]。

3. 在 [狀態機器] 清單中，選取您先前建立的狀態機器 (例如Helloworld)。
4. 保留頁面上的所有預設選項，然後選擇「下一步」。這會開啟 [設定標籤] 頁面。
5. 再次選擇 Next (下一步)。這會開啟 [檢閱並建立] 頁面。
6. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

   規則隨即建立並顯示「規則」頁面，列出您的所有 Amazon EventBridge 規則。

步驟 4：測試 規則

現在一切都已就緒，請測試將檔案新增至 Amazon S3 儲存貯體，然後查看產生的狀態機器執行輸入。

1. 將檔案新增至您的 Amazon S3 儲存貯體。

   導覽至 Amazon S3 主控台，選擇您建立用來存放檔案的儲存貯體 (*username*-sfn-tutorial)，然後選擇「上傳」。
2. 例如，新增檔案*test.png*，然後選擇 [上傳]。

   這麼做會啟動執行狀態機器，傳遞 AWS CloudTrail 中的資訊做為輸入。
3. 檢查狀態機器的執行狀況。

   導覽至 Step Functions 主控台，然後選取 Amazon EventBridge 規則中使用的狀態機器 (Helloworld)。
4. 選取該狀態機器的最近執行，然後展開 [執行輸入] 區段。

   此輸入包含儲存貯體名稱和物件名稱等資訊。在真實世界使用案例中，狀態機器可以使用此輸入對該物件執行動作。

執行輸入的範例

下面的例子顯示了一個典型的輸入到狀態機執行。

```
{
    "version": "0",
    "id": "6c540ad4-0671-9974-6511-756fbd7771c3",
    "detail-type": "Object Created",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2023-06-23T23:45:48Z",
```

```
    "region": "us-east-2",
    "resources": [
      "arn:aws:s3:::username-sfn-tutorial"
    ],
    "detail": {
      "version": "0",
      "bucket": {
        "name": "username-sfn-tutorial"
      },
      "object": {
        "key": "test.png",
        "size": 800704,
        "etag": "f31d8546bb67845b4d3048cde533b937",
        "sequencer": "00621049BA9A8C712B"
      },
      "request-id": "79104EXAMPLEB723",
      "requester": "123456789012",
      "source-ip-address": "200.0.100.11",
      "reason": "PutObject"
    }
  }
```

使用 API Gateway 建立 Step Functions 式 API

您可以使用 Amazon API Gateway 將您的 API 與 AWS Step Functions API Gateway API 中的方法建立關聯。當 HTTPS 要求傳送至 API 方法時，API Gateway 會叫用您的 Step Functions API 動作。

此教學課程說明如何使用一個資源和 POST 方法來建立 API，從而與 `StartExecution` API 動作通訊。您將使用 AWS Identity and Access Management (IAM) 主控台建立 API Gateway 的角色。然後，您將使用 API Gateway 主控台來建立 API Gateway API、建立資源和方法，並將方法對應至 `StartExecution` API 動作。最後，您將部署和測試 API。

> **Note**
>
> 雖然 Amazon API Gateway 可以透過呼叫`DescribeExecution`來啟動 Step Functions 式執行`StartExecution`，但您必須呼叫以取得結果。

主題

- 步驟 1：建立 API Gateway 的 IAM 角色

- 步驟 2：建立您的 API Gateway API
- 步驟 3：測試和部署 API Gateway API

步驟 1：建立 API Gateway 的 IAM 角色

在建立 API Gateway API 之前，您需要授與 API Gateway 權限，才能呼叫 Step Functions API 動作。

若要設定 API Gateway 的權限

1. 登入 IAM 主控台，然後選擇 [角色]、[建立角色]。
2. 在 Select trusted entity (選取信任的實體) 頁面上，執行以下作業：
   a. 對於信任的實體類型，請保留的預設選取項目AWS 服務。
   b. 對於使用案例，請從下拉式清單中選擇「API Gateway」。
3. 選取 [API Gateway]，然後選擇 [下一步]。
4. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。
5. (選擇性) 在名稱、檢閱和建立頁面上，輸入詳細資訊，例如角色名稱。例如，輸入 **`APIGatewayToStepFunctions`**。
6. 選擇建立角色。

   IAM 角色會顯示在角色清單中。
7. 選擇您角色的名稱，並記下 Role ARN (角色 ARN)，如以下範例所示。

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```

將政策附加到 IAM 角色

1. 在 Roles (角色) 頁面上，搜尋您的角色 (`APIGatewayToStepFunctions`)，然後選擇角色。
2. 在 [權限] 索引標籤上，選擇 [新增權限]，然後選擇 [附加原則]。
3. 在 [附加原則] 頁面上，搜尋`AWSStepFunctionsFullAccess`並選擇原則，然後選擇 [新增權限]。

步驟 2：建立您的 API Gateway API

建立 IAM 角色後，您可以建立自訂 API Gateway API。

若要建立 API

1. 開啟 Amazon API Gateway 主控台，然後選擇「建立 API」。
2. 在 [選擇 API 類型] 頁面的 [REST API] 窗格中，選擇 [建置]。
3. 在 [建立其餘 API] 頁面上，選取 [新增 API]，然後輸入 ***StartExecutionAPI*** 做為 API 名稱。
4. 將 API 端點類型保持為「地區」，然後選擇「建立 API」。

建立資源

1. 在 ***StartExecutionAPI*** 的 [資源] 頁面上，選擇 [建立資源]。
2. 在 [建立資源] 頁面上，輸入 [**execution**資源名稱]，然後選擇 [建立資源]。

若要建立 POST 方法

1. 選擇 /執行資源，然後選擇 [建立方法]。
2. 選擇做為「方法」類型POST。
3. 針對「整合類型」選擇「AWS 服務」。
4. 對於 AWS 區域，從清單中選擇「區域」。

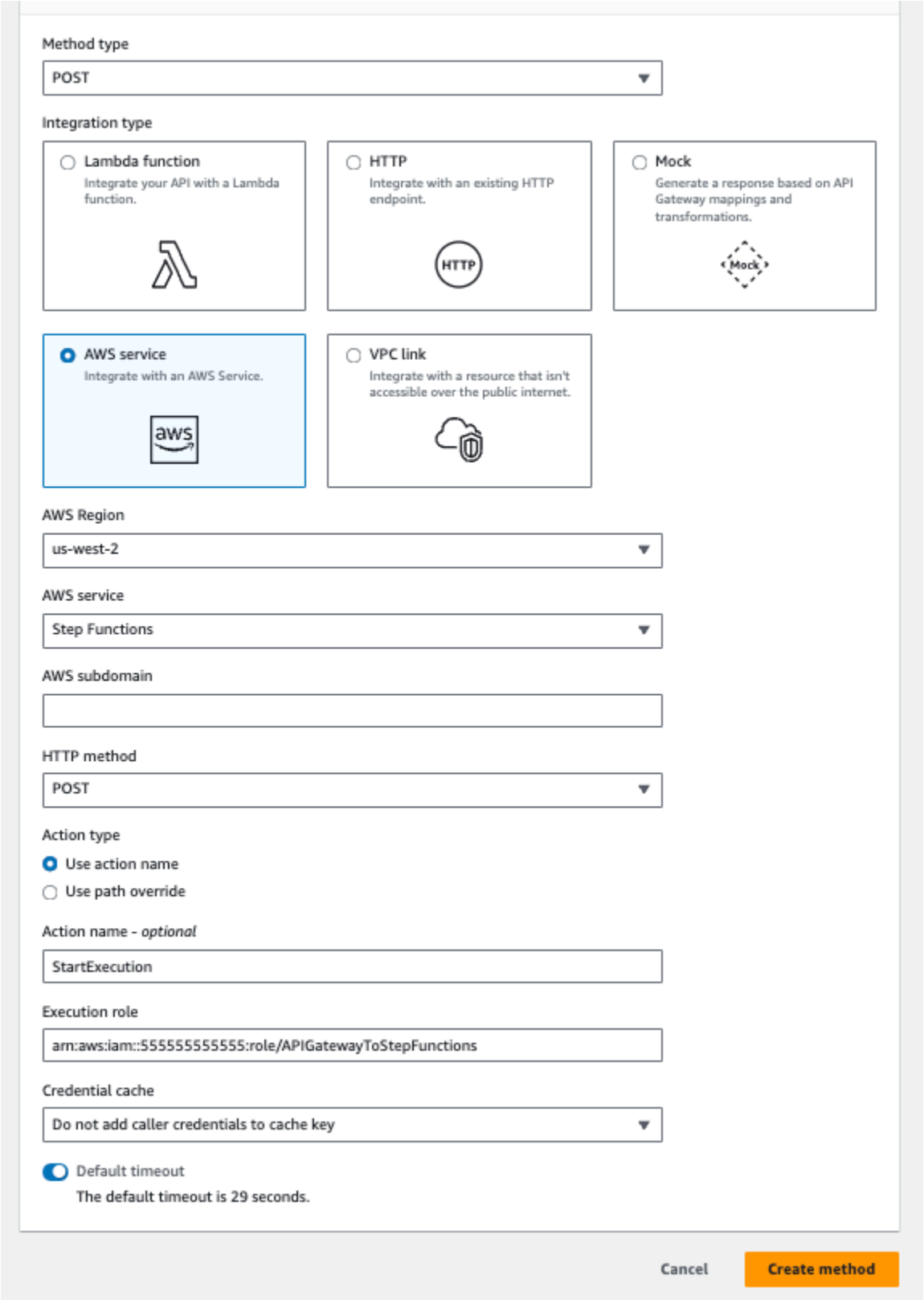
> **Note**
> 對於目前支援 Step Functions 的區域，請參閱支援的區域。

5. 對於 AWS 服務，從清單中選擇 Step Functions。
6. 讓 AWS 子網域保持空白。
7. 對於 HTTP 方法，請從清單中選擇 POST。

> **Note**
> 所有 Step Functions API 動作都使用 HTTP POST 方法。

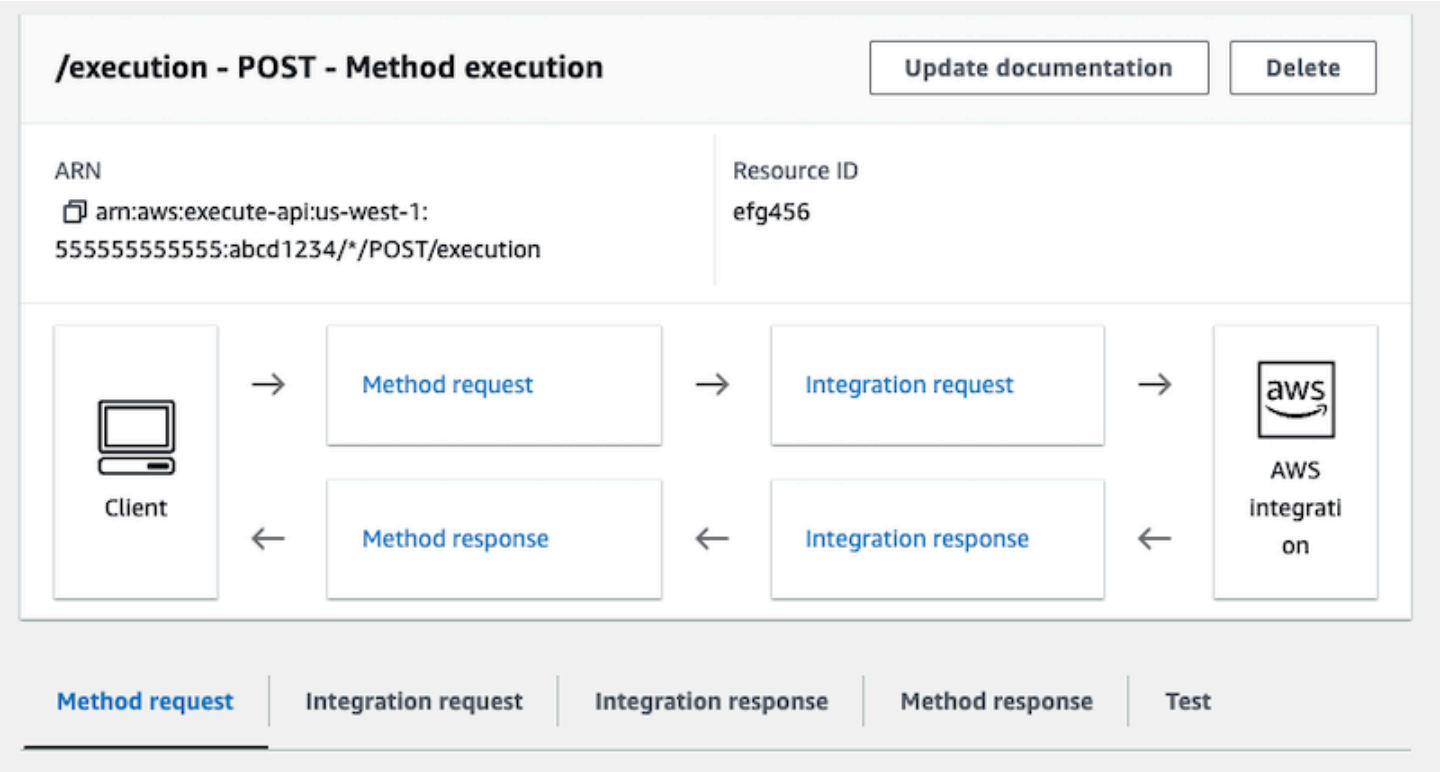
8. 針對動作類型，選取使用動作名稱。
9. 針對動作名稱，輸入 **StartExecution**。
10. 針對執行角色，輸入您先前建立之 IAM 角色的角色 ARN，如下列範例所示。

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```



11. 保留 [認證快取] 和 [預設逾時] 的預設選項，然後選擇 [儲存]。

API Gateway 和 Step Functions 之間的視覺對應會顯示在 /執行-POST-方法執行頁面上。



步驟 3：測試和部署 API Gateway API

建立 API 之後，請測試和部署它。

若要測試 API Gateway 與 Step Functions 之間的通訊

1. 在 /執行-POST-[方法執行] 頁面上，選擇 [測試] 索引標籤。您可能需要選擇向右箭頭按鈕才能顯示此索引標籤。
2. 在 /執行-POST-方法測試索引標籤上，使用現有狀態機器的 ARN (或建立使用 Lambda 函數的新狀態機器)，將下列要求參數複製到 [要求主體] 區段中，然後選擇 [測試]。

   ```
   {
      "input": "{}",
      "name": "MyExecution",
      "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
   }
   ```

   如需詳細資訊，請參閱 AWS Step Functions API 參考資料中的`StartExecution`要求語法。

> **Note**
>
> 如果您不想在 API Gateway 呼叫的主體中包含狀態機器的 ARN，您可以在整合要求索引標籤中設定對應範本，如下列範例所示。
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> 使用這種方法，您可以根據開發階段 (例如、、dev和prod) 指定不同狀態機器的 ARN。test如需有關在對應範本中指定階段變數的詳細資訊，請參閱 API Gateway 開發人員指南*$stageVariables*中的。

3. 執行開始，執行 ARN 及其紀元日期顯示在響應主體下。

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> 您可以在 AWS Step Functions 主控台上選擇您的狀態機器來檢視執行。

部署 API

1. 在 ***StartExecutionAPI*** 的 [資源] 頁面上，選擇 [部署 API]。
2. 針對階段，選取新階段。
3. 針對階段名稱，輸入 **alpha**。
4. 在描述，請輸入描述。
5. 選擇部署。

測試您的部署

1. 在 ***StartExecutionAPI*** 的「階段」頁面上，展開 Alpha、/、/執行、POST，然後選擇 POST 方法。
2. 在「方法覆寫」 下，選擇複製圖示以複製 API 的叫用 URL。完整的 URL 看起來應如下列範例所示。

```
https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

3. 從命令列使用您狀態機器的 ARN 執行 `curl` 命令，然後呼叫您部署的 URL，如以下範例所示。

```
curl -X POST -d '{"input": "{}","name": "MyExecution","stateMachineArn":
 "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"}' https://
a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

隨即傳回執行 ARN 及其 Epoch 日期，如以下範例所示。

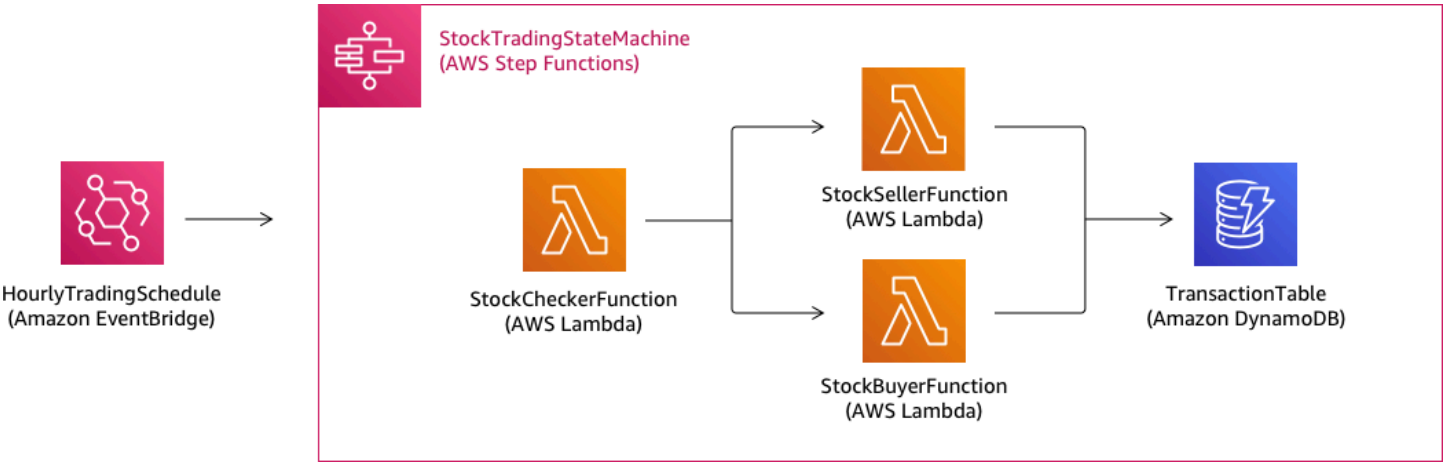
```
{"executionArn":"arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
```

> **Note**
>
> 如果您收到「缺少身份驗證令牌」錯誤，請確保調用 URL 以 /execution 結尾。

使用創建 Step Functions 狀態機AWS SAM

在本指南中，您會下載、建置和部署包含 AWS Step Functions 狀態機器的範例 AWS SAM 應用程式。這個應用程式會建立一個模擬股票交易工作流程，並依預先定義的排程執行 (請注意，排程預設為停用以避免產生費用)。

下圖顯示此應用程式的組件：



以下是您要建立範例應用程式所執行的命令預覽。如需當中各個命令的詳細資訊，請參閱此頁面稍後的章節

```
# Step 1 - Download a sample application. For this tutorial you
#   will follow the prompts to select an AWS Quick Start Template
#   called 'Multi-step workflow'
sam init

# Step 2 - Build your application
cd project-directory
sam build

# Step 3 - Deploy your application
sam deploy --guided
```

先決條件

本指南假設您已完成安裝 AWS SAM CLI 中針對您的作業系統的步驟。它假設您已完成以下操作：

1. 創建了AWS帳戶。
2. 已設定的 IAM 許可。
3. 已安裝 Homebrew。注意：Homebrew 只是 Linux 和 macOS 的先決條件。
4. 已安裝 AWS SAM CLI。注意：請確認您擁有 0.52.0 版或更高版本。您可以透過執行命令 `sam --version` 檢查您所擁有的版本。

步驟 1：下載範例 AWS SAM 應用程式

要執行的命令：

```
sam init
```

依照螢幕上的提示選取下列項目：

1. 模板： AWS快速入門範本
2. 語言：Python、Ruby、NodeJS、Go、Java 或 .NET
3. 專案名稱：(您選擇的名稱 - 預設為 sam-app)
4. 快速啟動應用程式：多步驟工作流程

什麼AWS SAM正在做：

此命令會使用您為「專案名稱」提示所提供的名稱來建立目錄 (預設為 sam-app)。目錄的特定內容將取決於您選擇的語言。

以下是當您選擇其中一個 Python 執行時間時的目錄內容：

```
### README.md
### functions
#   ### __init__.py
#   ### stock_buyer
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### stock_checker
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### stock_seller
#       ### __init__.py
#       ### app.py
#       ### requirements.txt
### statemachine
#   ### stock_trader.asl.json
### template.yaml
### tests
    ### unit
```

```
        ### __init__.py
        ### test_buyer.py
        ### test_checker.py
        ### test_seller.py
```

您可以查看兩個特別有趣的檔案：

- `template.yaml`：包含AWS SAM定義應用程式的範本AWS資源。
- `statemachine/stockTrader.asl.json`：包含應用程式的狀態機器定義，這是以 Amazon States Language 撰寫。

您可以在 `template.yaml` 檔案中看到下列項目，這是指向狀態機器定義檔：

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

將狀態機定義保留為單獨的文件，而不是將其嵌入AWS SAM範本。例如，如果您未在範本中包含定義，則追蹤狀態機定義的變更會比較容易。您可以使用 Workflow Studio 建立和維護狀態機器定義，並將定義從主控台直接匯出到 Amazon 州語言規格檔案，而不需將其合併到範本中。

如需範例應用程式的詳細資訊，請參閱專案目錄中的 `README.md` 檔案。

步驟 2：建置您的應用程式

要執行的命令：

首先變更至專案目錄 (也就是範例應用程式的 `template.yaml` 檔案所在的目錄；依預設為 `sam-app`)，然後執行下列命令：

```
sam build
```

示例輸出：

```
Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml
```

```
Commands you can use next
=========================
[*] Invoke Function: sam local invoke
[*] Deploy: sam deploy --guided
```

什麼AWS SAM正在做：

該AWS SAMCLI 隨附許多 Lambda 執行階段的抽象，可用來建置相依性，並將所有組建成品複製到暫存資料夾中，以便可以封裝和部署所有項目。`sam build` 命令會建置應用程式所具有的任何相依性，並將建置成品複製到 `.aws-sam/build` 下的資料夾。

步驟 3：將應用程式部署至AWS雲端

要執行的命令：

```
sam deploy --guided
```

依照螢幕上的提示操作。您可以直接以 `Enter` 回應以接受互動體驗中提供的預設選項。

什麼AWS SAM正在做：

此命令將您的應用程式部署至AWS雲。它需要您使用構建的部署工件`sam build`命令、封裝並將其上傳至 Amazon S3 儲存貯體建立的形式上傳至 Amazon S3AWS SAMCLI，並使用以下方式部署應用程式AWS CloudFormation。在部署命令的輸出中，您可以看到對 AWS CloudFormation 堆疊進行的變更。

您可以按照以下步驟驗證示例 Step Functions 狀態機器已成功部署：

1. 請將物件上傳至AWS Management Console並在以下位置開啟 Step Functions 主控台https://console.aws.amazon.com/states/。
2. 在左側導覽中，選擇 State machines (狀態機器)。
3. 在清單中尋找並選擇您的新狀態機器。它將被命名 StockTradingStateMachine-*<unique-hash>*。
4. 選擇 Definition (定義) 索引標籤。

您現在應該可以看到狀態機器的視覺表示。您可以驗證視覺表示是否與專案目錄的 `statemachine/stockTrader.asl.json` 檔案中找到的狀態機器定義相符。

疑難排解

SAM CLI 錯誤：「沒有此類選項：--guided」

執行 `sam deploy` 時，您會看到下列錯誤：

```
Error: no such option: --guided
```

這表示您正在使用不支援 `--guided` 參數的舊版 AWS SAM CLI。若要修正此問題，您可以將 AWS SAM CLI 版本更新為 0.33.0 或更新版本，或從 `sam deploy` 命令中省略 `--guided` 參數。

SAM CLI 錯誤:「無法建立受管資源：找不到登入資料」

執行 `sam deploy` 時，您會看到下列錯誤：

```
Error: Failed to create managed resources: Unable to locate credentials
```

這表示您尚未設定AWS認證以啟用AWS SAM要製作的 CLIAWS服務電話。若要解決此問題，您必須設定AWS認證。如需詳細資訊，請參閱正在設定AWS認證在AWS Serverless Application Model開發者指南。

清除

如果您不再需要AWS您透過執行此教學課程建立的資源，您可以刪除AWS CloudFormation您部署的堆疊。

若要刪除透過此教學使用AWS Management Console所建立的 AWS CloudFormation 堆疊，請依照下列步驟執行：

1. 請登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/cloudformation 的 AWS CloudFormation 主控台。
2. 在左側導覽窗格中，選擇 Stacks (堆疊)。
3. 在堆疊清單中，選擇 sam-app (或您建立的堆疊名稱)。
4. 選擇 刪除 。

完成後，堆疊的狀態將變更為 DELETE_COMPLETE。

或者，您可以執行下列 AWS CLI 命令來刪除 AWS CloudFormation 堆疊：

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

驗證已刪除的堆疊

對於這兩種刪除方法AWS CloudFormation堆棧，您可以通過轉到以驗證它已刪除https://console.aws.amazon.com/cloudformation，選擇堆疊在左側導航窗格中，然後選擇已刪除在搜尋文字方塊右側的下拉式選單中。您應該在刪除的堆疊清單中看到堆疊名稱 sam-app (或您所建立的堆疊名稱)。

使用 Step Functions 創建活動狀態機

本教學課程會說明如何使用 Java 和 AWS Step Functions，建立以活動為基礎的狀態機器。活動可讓您控制從狀態機器執行的其他位置的 Worker 程式碼。如需概觀，請參閱 Step Functions 的工作原理中的 活動。

為了完成本教學，您需要以下項目：

- 適用於 Java 的 開發套件。本教學課程中的範例活動是一個 Java 應用程式，它使用與通訊 AWS。AWS SDK for Java
- AWS 環境或標準 AWS 組態檔案中的認證。如需詳細資訊，請參閱AWS SDK for Java 開發人員指南中的「設定 AWS 認證」。

主題

步驟 1：建立活動

您必須讓 Step Functions 知道您要建立其 Worker (程式) 的活動。Step Functions 會使用建立活動身分的 Amazon 資源名稱 (ARN) 回應。使用此身分來協調狀態機器和工作者之間傳遞的資訊。

> **Important**
>
> 確保您的活動任務與狀態機位於相同的 AWS 帳戶下。

1. 在「Step Functions」主控台的左側導覽窗格中，選擇「活動」。
2. 選擇 Create activity (建立活動)。
3. 輸入活動的 [名稱]，例如*get-greeting*，然後選擇 [建立活動]。
4. 建立活動任務時，請記下其 ARN，如以下範例所示。

```
arn:aws:states:us-east-1:123456789012:activity:get-greeting
```

步驟 2：建立狀態機

建立狀態機器以決定何時呼叫您的活動，以及您的工作者何時應執行其主要工作、收集並傳回結果。若要建立狀態機，您將使用程式碼編輯器的工作流程工作室。

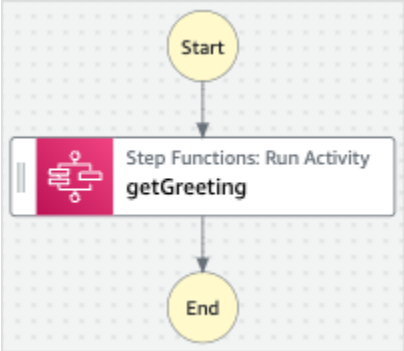
1. 在 [Step Functions] 主控台的左側導覽窗格中，選擇 [狀態機器]。
2. 在 [狀態機器] 頁面上，選擇 [建立狀態機器]。
3. 在「選擇範本」對話方塊中，選取「空白」。
4. 選擇選取。這會在設計模式中開啟工作流程工作室
5. 在本教程中，您將在代碼編輯器中編寫狀態機的Amazon States Language（ASL）定義。若要這麼做，請選擇 [程式碼]。
6. 刪除現有的樣板代碼並粘貼以下代碼。請記得將此程式碼中的範例 ARN 取代為您先前在欄位中建立的活動任務的 `Resource` ARN。

```
{
  "Comment": "An example using a Task state.",
  "StartAt": "getGreeting",
  "Version": "1.0",
  "TimeoutSeconds": 300,
  "States":
  {
    "getGreeting": {
      "Type": "Task",
      "Resource": "arn:aws:states:us-east-1:123456789012:activity:get-greeting",
```

```
      "End": true
    }
  }
}
```

這是使用Amazon States Language（ASL）對狀態機的描述。它定義名為 `getGreeting` 的單一 `Task` 狀態。如需詳細資訊，請參閱狀態機器結構。

7. 在上圖形視覺化窗格，請確定您新增的 ASL 定義的工作流程圖看起來與下圖類似。



8. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **`ActivityStateMachine`**。

9. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器設定中的所有預設選項。

   如果您先前已使用狀態機器的正確許可建立 IAM 角色，並且想要使用它，請在 [權限] 中選取 [選擇現有角色]，然後從清單中選取角色。或選取 [輸入角色 ARN]，然後為該 IAM 角色提供 ARN。

10. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

    您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 3：實作工作者

建立工作者。工作者是一種程式，負責：

- 輪詢使用 GetActivityTask API 動作之活動的 Step Functions。
- 使用您的程式碼執行活動工作 (例如，以下程式碼中的 getGreeting( ) 方法)。
- 使用 SendTaskSuccess、SendTaskFailure 和 SendTaskHeartbeat API 動作傳回結果。

> **Note**
>
> 如需活動工作者的更完整範例，請參閱 Ruby 中的範例活動工作者。此範例根據最佳實務提供實作，可供您的活動工作者參考。此程式碼會實作可為輪詢器和活動工作者設定執行緒數量的消費者-生產者模式。

實作工作者

1. 建立名為 GreeterActivities.java 的檔案。
2. 新增以下程式碼至其中。

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.auth.EnvironmentVariableCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.stepfunctions.AWSStepFunctions;
import com.amazonaws.services.stepfunctions.AWSStepFunctionsClientBuilder;
import com.amazonaws.services.stepfunctions.model.GetActivityTaskRequest;
import com.amazonaws.services.stepfunctions.model.GetActivityTaskResult;
import com.amazonaws.services.stepfunctions.model.SendTaskFailureRequest;
import com.amazonaws.services.stepfunctions.model.SendTaskSuccessRequest;
import com.amazonaws.util.json.Jackson;
import com.fasterxml.jackson.databind.JsonNode;
import java.util.concurrent.TimeUnit;


public class GreeterActivities {

    public String getGreeting(String who) throws Exception {
        return "{\"Hello\": \"" + who + "\"}";
    }
```

```
    public static void main(final String[] args) throws Exception {
        GreeterActivities greeterActivities = new GreeterActivities();
        ClientConfiguration clientConfiguration = new ClientConfiguration();
        clientConfiguration.setSocketTimeout((int)TimeUnit.SECONDS.toMillis(70));

        AWSStepFunctions client = AWSStepFunctionsClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
                .withCredentials(new EnvironmentVariableCredentialsProvider())
                .withClientConfiguration(clientConfiguration)
                .build();

        while (true) {
            GetActivityTaskResult getActivityTaskResult =
                    client.getActivityTask(
                            new
 GetActivityTaskRequest().withActivityArn(ACTIVITY_ARN));

            if (getActivityTaskResult.getTaskToken() != null) {
                try {
                    JsonNode json =
 Jackson.jsonNodeOf(getActivityTaskResult.getInput());
                    String greetingResult =

 greeterActivities.getGreeting(json.get("who").textValue());
                    client.sendTaskSuccess(
                            new SendTaskSuccessRequest().withOutput(

 greetingResult).withTaskToken(getActivityTaskResult.getTaskToken()));
                } catch (Exception e) {
                    client.sendTaskFailure(new
 SendTaskFailureRequest().withTaskToken(
                            getActivityTaskResult.getTaskToken()));
                }
            } else {
                Thread.sleep(1000);
            }
        }
    }
}
```

> **Note**
>
> 此範例中的 `EnvironmentVariableCredentialsProvider` 類別假設已設定 `AWS_ACCESS_KEY_ID` (或 `AWS_ACCESS_KEY`) 和 `AWS_SECRET_KEY` (或 `AWS_SECRET_ACCESS_KEY`) 環境變數。如需有關向工廠提供所需認證的詳細資訊，請參閱《開發AWS SDK for Java 人員指南》AWSCredentialsProvider中的「AWS SDK for Java API 參考」和「設定 AWS 認證和開發區域」中的。
> 根據預設， AWS SDK 最多會等待 50 秒，從伺服器接收資料以進行任何作業。此 `GetActivityTask` 操作是長時間輪詢操作，將等待最多 60 秒，以取得下一個可用任務。若要避免收到`SocketTimeoutException`錯誤訊息， SocketTimeout 請設定為 70 秒。

3. 在 `GetActivityTaskRequest().withActivityArn()` 建構函數的參數清單，將 `ACTIVITY_ARN` 值取代為您先前建立的活動任務 ARN。

步驟 4：運行狀態機

當您開始執行狀態機器時，Worker 會輪詢 Step Functions 的活動，執行其工作 (使用您提供的輸入)，並傳回其結果。

1. 在***ActivityStateMachine***頁面上，選擇 [開始執行]。

   此時會顯示「開始執行」對話方塊。

2. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   b. 在 [輸入] 方塊中，輸入下列 JSON 輸入以執行您的工作流程。

```
{
```

```
  "who": "AWS Step Functions"
}
```

c. 選擇 Start execution (開始執行)。

d. 「Step Functions」主控台會將您引導至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

步驟 5：執行和停用工作者

若要讓工作者輪詢您的活動狀態機器，您必須執行工作者。

1. 在命令列上，導覽至您建立 `GreeterActivities.java` 的目錄。
2. 要使用 AWS SDK，請將`lib`和`third-party`目錄的完整路徑添加到構建文件的依賴項和 Java 中`CLASSPATH`。如需詳細資訊，請參閱AWS SDK for Java 開發人員指南中的下載和解壓縮 SDK。
3. 編譯檔案。

```
$ javac GreeterActivities.java
```

4. 執行 檔案。

```
$ java GreeterActivities
```

5. 在「Step Functions」主控台上，瀏覽至「執行詳細資訊」頁面。
6. 執行完成時，請檢查執行結果。
7. 停用工作者。

用 Lambda 迭代一個循環

在此教學課程中，您將實作設計模式，其會使用狀態機器和 AWS Lambda 函數反覆運算迴圈特定的次數。

每當您需要追蹤狀態機器中的迴圈次數時，請使用此設計模式。此實作可協助您中斷大型任務，或將長時間執行的執行分割成較小區塊，或在特定事件數量後結束執行。您可以使用類似的實作來定期結束並

重新啟動長時間執行的執行，以避免超過 AWS Step Functions AWS Lambda、或其他服務的服 AWS 務配額。

在開始之前，請先閱讀建立使用 Lambda 的 Step Functions 狀態機器教學課程，以確保您熟悉一起使用 Lambda 和 Step Functions。

步驟 1：創建一個 Lambda 函數以迭代計數

透過使用 Lambda 函數，您可以追蹤狀態機器中迴圈的迭代次數。下列 Lambda 函數會接收`count`、`index`和的輸入值`step`。它會傳回包含更新 `index` 和名為 `continue` 布林值的這些值。如果小於，`true`則 Lambda 函數會`index`設定`continue`為`count`。

您的狀態機器會實作 `Choice` 狀態，如果 `continue` 為 `true`，其將執行一些應用程式邏輯，如果為 `false` 則結束。

建立 Lambda 函數

1. 登入 Lambda 主控台，然後選擇 [建立函數]。
2. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。
3. 在「基本資訊」區段中，設定 Lambda 函數，如下所示：
   a. 針對 函數名稱 ，請輸入 `Iterator`。
   b. 針對 Runtime (執行時間)，選擇 Node.js 。
   c. 在 [變更預設執行角色] 中，選擇 [使用基本 Lambda 權限建立新角色]。
   d. 選擇建立函數。
4. 將 Lambda 函數的下列程式碼複製到程式碼來源中。

```
export const handler = function (event, context, callback) {
  let index = event.iterator.index
  let step = event.iterator.step
  let count = event.iterator.count

  index = index + step

  callback(null, {
    index,
    step,
    count,
    continue: index < count
  })
```

```
}
```

此程式碼接受輸入值為 count、index 和 step。這會將 step 的值增量 index，並傳回這些值及布林值 continue。如果 index 低於 count，則 continue 的值為 true。

5. 選擇部署。

步驟 2：測試 Lambda 函數

使用數值執行 Lambda 函數，以便在運作中查看該函數。您可以為模擬迭代的 Lambda 函數提供輸入值。

若要測試您的 Lambda 函數

1. 選擇 測試。
2. 在 [設定測試事件] 對話方塊TestIterator中，輸入 [事件名稱] 方塊。
3. 將範例資料取代為以下內容。

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

這些值會模擬反覆運算期間來自狀態機器的內容。Lambda 函數將遞增索引，並在索引小true於continue時返回count。對於此測試，索引已經增加到 5。測試將遞增index至6並設定continue為true。

4. 選擇建立。
5. 選擇 [測試] 以測試您的 Lambda 函數。

測試結果會顯示在 [執行結果] 索引標籤中。

6. 選擇 [執行結果] 索引標籤以查看輸出。

```
{
  "index": 6,
  "step": 1,
```

```
  "count": 10,
  "continue": true
}
```

> **Note**
>
> 如果您將設index定為9並再次測試10，則continue會index增量為和false。

步驟 3：建立狀態機器

> **離開 Lambda 主控台之前...**
>
> 複製 Lambda 函數 ARN。將其粘貼到註釋中。下一個步驟將需要此值。

接下來，您將建立具有下列狀態的狀態機器：

- ConfigureCount— 設定count、index和的預設值step。
- Iterator— 參照您之前建立的 Lambda 函數，並傳入中設定的值ConfigureCount。
- IsCountReached— 根據Iterator函數傳回的值，繼續迴圈或繼續Done狀態的選擇狀態。
- ExampleWork— 需要完成工作的存根。在此範例中，工作流程具有Pass狀態，但在實際解決方案中，您可能會使用Task.
- Done— 工作流程的結束狀態。

若要在主控台中建立狀態機器：

1. 開啟 Step Functions 主控台，然後選擇 [建立狀態機器]。

   > **Important**
   >
   > 您的狀態機器必須與 Lambda 函數位於相同的 AWS 帳戶和區域中。

2. 選取「空白」範本。
3. 在「程式碼」窗格中，貼上下列定義狀態機器的 JSON。

   如需 Amazon 州語言的詳細資訊，請參閱狀態機器結構。

```
{
    "Comment": "Iterator State Machine Example",
    "StartAt": "ConfigureCount",
    "States": {

        "ConfigureCount": {
            "Type": "Pass",
            "Result": {
                "count": 10,
                "index": 0,
                "step": 1
            },
            "ResultPath": "$.iterator",
            "Next": "Iterator"
        },
        "Iterator": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Iterate",
            "ResultPath": "$.iterator",
            "Next": "IsCountReached"
        },
        "IsCountReached": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.iterator.continue",
                    "BooleanEquals": true,
                    "Next": "ExampleWork"
                }
            ],
            "Default": "Done"
        },
        "ExampleWork": {
            "Comment": "Your application logic, to run a specific number of times",
            "Type": "Pass",
            "Result": {
              "success": true
            },
            "ResultPath": "$.result",
            "Next": "Iterator"
        },
        "Done": {
            "Type": "Pass",
```

```
            "End": true

        }
    }
}
```

4. 將`Iterator Resource`欄位取代為您先前建立的 `Iterator` Lambda 函數的 ARN。
5. 選取「Config」，然後輸入狀態機器的「名稱」，例如*`IterateCount`*。

> **Note**
>
> 狀態機器、執行項目和活動工作的名稱長度不得超過 80 個字元。這些名稱對於您的帳戶和 AWS 地區而言必須是唯一的，且不得包含以下任何一項：
>
> - 空白
> - 萬用字元 (? *)
> - 括號字元 (< > { } [ ])
> - 特殊字元 (: ; , \ | ^ ~ $ # % & ` ")
> - 控制字符 (\\u0000-\\u001f 或 \\u007f-\\u009f)。
>
> 如果您的狀態機器的類型為 Express，則可以為狀態機器的多個執行提供相同的名稱。即使多個執行具有相同的名稱，Step Functions 也會為每個 Express 狀態機器執行產生唯一的執行 ARN。
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

6. 對於「類型」，接受預設值為「標準」。針對「權限」，選擇「建立新角色」。
7. 選擇 [建立]，然後選擇 [確認角色建立]。

步驟 4：開始新的執行

建立狀態機器後，您就可以開始執行。

1. 在IterateCount頁面上，選擇 [開始執行]。
2. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

3. 選擇 Start Execution (開始執行)。

   狀態機器的新執行隨即開始，顯示您正在執行的執行。
   狀態機器圖形視圖，以藍色顯示迭代器狀態，以指示進行中的狀態。

   執行會以步驟遞增，並使用 Lambda 函數追蹤計數。每次反覆運算時，其會執行在狀態機器中以 `ExampleWork` 狀態參考的範例工作。

   計數達到狀態機器中指定的 `ConfigureCount` 狀態的數量後，執行會結束反覆運算並退出。
   狀態機器圖形檢視，以綠色顯示 [迭代器] 狀態和 [完成] 狀態，表示兩者都已成功。

將長時間執行的工作流程執行作為新的執行作業

AWS Step Functions 旨在執行具有有限持續時間和步驟數量的工作流程。執行的持續時間上限為一年，最多可以有 25,000 個事件 (請參閱 配額)。

對於長時間執行的執行，為了避免在執行事件歷史記錄中達到 25,000 個項目的硬配額，我們建議您直接從狀態機器的`Task`狀態開始新的工作流程執行。這可讓您將工作流程分解為較小的狀態機器，並在新的執行中繼續進行中的工作。若要啟動這些工作流程執行，請從您的`Task`狀態呼叫 `StartExecution` API 動作，並傳遞必要的參數。

或者，您也可以實作模式，該模式使用 Lambda 函數來啟動狀態機器的新執行，以將正在進行的工作分割為多個工作流程執行。

本教學課程將說明在不超過服務配額的情況下繼續工作流程執行的兩種方法。

主題

- 使用 Step Functions API 動作繼續新的執行 (建議)
- 使用 Lambda 函數繼續新的執行

使用 Step Functions API 動作繼續新的執行 (建議)

Step Functions 可以透過呼叫自己的 API 做為整合式服務來啟動工作流程執行。我們建議您使用此方法來避免長時間執行的執行超出服務配額。

步驟 1：建立長時間執行的狀態機器

建立要從不同狀態機器狀態啟動的長時間執行Task狀態機器。在本教學課程中，請使用使用 Lambda 函數的狀態機器。

> **Note**
>
> 請務必將此狀態機器的名稱和 Amazon 資源名稱複製到文字檔中，以供日後使用。

步驟 2：建立狀態機以呼叫 Step Functions 式 API 動作

若要從狀態啟動工作流程執行 **Task**

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 從「動作」索引標籤中，將 StartExecutionAPI 動作拖放至標示為「拖曳第一個狀態」的空白狀態。
5. 選擇狀StartExecution態並在中的「組態」索引標籤中執行下列動作設計模式：

    a. 將狀態重新命名為**`Start nested execution`**。

    b. 針對 [整合類型]，從下拉式清單中選擇 [AWS SDK-新增]。

    c. 在 API 參數中，執行下列動作：

        i. 對於`StateMachineArn`，將範例 Amazon 資源名稱取代為狀態機器的 ARN。例如，輸入使用 Lambda 之狀態機器的 ARN。

        ii. 若為`Input`節點，請使用下列值取代現有的預留位置文字：

        ```
        "Comment": "Starting workflow execution using a Step Functions API action"
        ```

        iii. 確保您在 API 參數中的輸入看起來類似於以下內容：

```
{
  "StateMachineArn": "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine",
  "Input": {
    "Comment": "Starting workflow execution using a Step Functions API
 action",
    "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
  }
```

6. (選擇性) 選擇Inspector面板上的「定義」，以檢視工作流程的自動產生 Amazon States Language (ASL) 定義。

   > **Tip**
   >
   > 您也可以在工作流程工作室中檢視 ASL 定義。程式碼編輯器在程式碼編輯器中，您也可以編輯工作流程的 ASL 定義。

7. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **`ParentStateMachine`**。

8. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器設定中的所有預設選項。

   如果您先前已使用狀態機器的正確許可建立 IAM 角色，並且想要使用它，請在 [權限] 中選取 [選擇現有角色]，然後從清單中選取角色。或選取 [輸入角色 ARN]，然後為該 IAM 角色提供 ARN。

9. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

   > **Note**
   >
   > 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 3：更新 IAM 政策

為了確保您的狀態機具有開始執行使用 Lambda 函數的狀態機器的許可，您需要將內嵌政策附加到狀態機的 IAM 角色。如需詳細資訊，請參閱 IAM 使用者指南中的內嵌內嵌政策。

1. 在ParentStateMachine頁面上，選擇 IAM 角色 ARN 以導覽至狀態機器的 IAM 角色頁面。
2. 將適當的權限指派給的 IAM 角色，讓它能夠開始執行另一個狀態機器。ParentStateMachine若要指派權限，請執行下列動作：

   a. 在 [IAM 角色] 頁面上，選擇 [新增權限]，然後選擇 [建立內嵌政策]。

   b. 在建立政策頁面上，選擇 JSON 標籤。

   c. 以下列原則取代現有文字。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
  ]
}
```

   d. 選擇檢閱政策。

   e. 指定策略的名稱，然後選擇 [建立策略]。

步驟 4：運行狀態機

狀態機器執行是執行工作流程以執行工作的執行個體。

1. 在ParentStateMachine頁面上，選擇 [開始執行]。

   此時會顯示「開始執行」對話方塊。

2. 在 [開始執行] 對話方塊中，執行下列動作：

a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

b. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

c. 選擇 Start execution (開始執行)。

d. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

3. 開啟LambdaStateMachine頁面並注意由觸發的新執行ParentStateMachine。

使用 Lambda 函數繼續新的執行

您可以建立使用 Lambda 函數的狀態機器，在目前的執行終止之前開始新的執行。使用此方法在新的執行中繼續進行中的工作，可讓您擁有可以將大型工作分解為較小工作流程的狀態機器，或者擁有無限期執行的狀態機器。

本教學課程以使用外部 Lambda 函數修改工作流程的概念為基礎，此概念已在用 Lambda 迭代一個循環教學課程中展示。您可以使用相同的 Lambda 函數 (`Iterator`) 來反覆執行特定次數的迴圈。此外，您還可以建立另一個 Lambda 函數來開始新的工作流程執行，並在每次啟動新執行時遞減計數。透過設定輸入中的執行次數，此狀態機器會結束並重新啟動指定次數的執行。

您將建立的狀態機器會實作下列狀態。

State	用途
`ConfigureCount`	一種`Pass`狀態`count`，可設定 `Iterator` Lambda 函數用來逐步執行工作迭代的`index`、和`step`值。

State	用途
`Iterator`	參考 `Iterator` Lambda 函數的`Task`狀態。
`IsCountReached`	使用`Iterator`函數中的 Boolean 值來決定狀態機是否應該繼續範例工作，還是移至`ShouldRestart` 狀態的狀態。`Choice`
`ExampleWork`	表示將在實際實施中執行工作的`Task`狀態的狀態。`Pass`
`ShouldRestart`	使用`executionCount` 值來決定是否應該結束某個執行並開始另一個執行，或直接結束的`Choice`狀態。
`Restart`	使用 Lambda 函數來啟動`Task`狀態機器的新執行的狀態。如同 `Iterator` 函數，此函數也會減少計數。狀`Restart`態將計數的遞減值傳遞給新執行的輸入。

必要條件

在開始之前，請先閱讀建立使用 Lambda 的 Step Functions 狀態機器教學課程，以確保您熟悉一起使用 Lambda 和 Step Functions。

主題

- 步驟 1：創建一個 Lambda 函數以迭代計數
- 步驟 2：建立重新啟動 Lambda 函數以開始執行新的 Step Functions 數
- 步驟 3：建立狀態機
- 步驟 4：更新 IAM 政策
- 步驟 5：運行狀態機

步驟 1：創建一個 Lambda 函數以迭代計數

> **Note**
>
> 如果您已完成用 Lambda 迭代一個循環教學課程，則可以略過此步驟並使用該 Lambda 函數。

本節和用 Lambda 迭代一個循環教學課程說明如何使用 Lambda 函數追蹤計數，例如狀態機器中迴圈的迭代次數。

下列 Lambda 函數會接收countindex、和的輸入值step。它會傳回包含更新 index 和名為 continue 布林值的這些值。如果小於，true則 Lambda 函數會index設定continue為count。

您的狀態機器實作 Choice 狀態，然後執行一些應用程式邏輯，如果 continue 是 true，或移至 ShouldRestart (如果 continue 是 false)。

建立反覆執行 Lambda 函數

1. 開啟 Lambda 主控台，然後選擇 Create function (建立函數)。
2. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。
3. 在「基本資訊」區段中，設定 Lambda 函數，如下所示：

   a. 針對 函數名稱 ，請輸入 Iterator。

   b. 針對 執行時間，請選擇 Node.js 16.x。

   c. 保留頁面上的所有預設選項，然後選擇 [建立功能]。

   建立 Lambda 函數時，請在頁面右上角記下其 Amazon 資源名稱 (ARN)，例如：

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

4. 將 Lambda 函數的下列程式碼複製到 Lambda 主控台中「#代器」頁面的「程式碼來源」區段。

   ```
   exports.handler = function iterator (event, context, callback) {
     let index = event.iterator.index;
     let step = event.iterator.step;
     let count = event.iterator.count;

     index = index + step;

     callback(null, {
       index,
       step,
       count,
       continue: index < count
     })
   }
   ```

   此程式碼接受輸入值為 count、index 和 step。這會將 step 的值增加 index，並傳回這些值及布林值 continue。如果 index 低於 count，則 continue 的值為 true。

5. 選擇部署以部署程式碼。

測試重複使用 Lambda 函數

若要查看您的 Iterate 函數運作狀況，請使用數值來執行該函數。您可以為模擬迭代的 Lambda 函數提供輸入值，以查看使用特定輸入值獲得的輸出。

若要測試您的 Lambda 函數

1. 在 [設定測試事件] 對話方塊中，選擇 [建立新測試事件]，然後輸入 TestIterator [事件名稱]。
2. 將範例資料取代為以下內容。

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

這些值會模擬反覆運算期間來自狀態機器的內容。Lambda 函數遞增索引並返回continue為true。當索引未低於 count，就會以 false 的形式傳回 continue。對於此測試，索引已經增加到 5。結果應該將 index 增加到 6 和將 continue 設定為 true。

3. 選擇建立。
4. 在 Lambda 主控台的 [###] 頁面上，確定TestIterator已列出，然後選擇 [測試]。

測試結果會顯示在頁面頂端。選擇 Details (詳細資訊) 並檢閱結果。

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> 如果您將此測試的 index 設定為 9，則 index 會增加至 10，且 continue 為 false。

步驟 2：建立重新啟動 Lambda 函數以開始執行新的 Step Functions 數

1. 開啟 Lambda 主控台，然後選擇 Create function (建立函數)。
2. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。
3. 在「基本資訊」區段中，設定 Lambda 函數，如下所示：
   a. 針對 函數名稱 ，請輸入 `Restart`。
   b. 針對 執行時間，請選擇 Node.js 16.x。
4. 保留頁面上的所有預設選項，然後選擇 [建立功能]。

   建立 Lambda 函數時，請在頁面右上角記下其 Amazon 資源名稱 (ARN)，例如：

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

5. 將 Lambda 函數的下列程式碼複製到 Lambda 主控台中「**####**」頁面的「程式碼來源」區段。

   下列程式碼會減少執行數量的計數，並啟動狀態機器的新執行，包括遞減數值。

   ```
   var aws = require('aws-sdk');
   var sfn = new aws.StepFunctions();

   exports.restart = function(event, context, callback) {

     let StateMachineArn = event.restart.StateMachineArn;
     event.restart.executionCount -= 1;
     event = JSON.stringify(event);

     let params = {
         input: event,
         stateMachineArn: StateMachineArn
     };

     sfn.startExecution(params, function(err, data) {
         if (err) callback(err);
         else callback(null,event);
     });

   }
   ```

6. 選擇部署以部署程式碼。

步驟 3：建立狀態機

現在您已經建立了兩個 Lambda 函數，請建立狀態機器。在此狀態機器中，ShouldRestart 和 Restart 狀態表示您在多次執行間中斷工作的方式。

Example ShouldRestart 選擇狀態

以下摘錄顯示狀ShouldRestartChoice態。此狀態決定您是否應該重新啟動執行。

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
  }
],
```

$.restart.executionCount 值包含在初始執行的輸入中。該值會在每次呼叫 Restart 函數時減 1，然後在每次後續執行時將該值放置於輸入中。

Example 重新啟動任務狀態

以下摘錄顯示狀RestartTask態。此狀態會使用您先前建立的 Lambda 函數來重新啟動執行，並遞減計數以追蹤剩餘要開始的執行次數。

```
"Restart": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

建立 狀態機器

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。

> **Important**
>
> 請確定您的狀態機與您先前在步驟 1 和步驟 2 中建立的 Lambda 函數位於相同的 AWS 帳戶和區域。

2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 在本教學課程中，您將在程式碼編輯器. Amazon States Language 要做到這一點，選擇代碼。
5. 刪除現有的樣板代碼並粘貼以下代碼。請記得將此程式碼中的 ARN 取代為您建立的 Lambda 函數的 ARN。

```
{
    "Comment": "Continue-as-new State Machine Example",
    "StartAt": "ConfigureCount",
    "States": {
        "ConfigureCount": {
            "Type": "Pass",
            "Result": {
                "count": 100,
                "index": -1,
                "step": 1
            },
            "ResultPath": "$.iterator",
            "Next": "Iterator"
        },
        "Iterator": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Iterator",
            "ResultPath": "$.iterator",
            "Next": "IsCountReached"
        },
        "IsCountReached": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.iterator.continue",
                    "BooleanEquals": true,
                    "Next": "ExampleWork"
                }
            ],
            "Default": "ShouldRestart"
        },
        "ExampleWork": {
            "Comment": "Your application logic, to run a specific number of times",
            "Type": "Pass",
            "Result": {
              "success": true
```

```
            },
            "ResultPath": "$.result",
            "Next": "Iterator"
        },
        "ShouldRestart": {
          "Type": "Choice",
          "Choices": [
            {
              "Variable": "$.restart.executionCount",
              "NumericGreaterThan": 0,
              "Next": "Restart"
            }
          ],
          "Default": "Done"
        },
        "Restart": {
          "Type": "Task",
          "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
          "Next": "Done"
        },
        "Done": {
            "Type": "Pass",
            "End": true
        }
    }
}
```

6. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **ContinueAsNew**。

7. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在本教學課程中，請保留狀態機器設定中的所有預設選項。

   如果您先前已使用狀態機器的正確許可建立 IAM 角色，並且想要使用它，請在 [權限] 中選取 [選擇現有角色]，然後從清單中選取角色。或選取 [輸入角色 ARN]，然後為該 IAM 角色提供 ARN。

8. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

9. 將此狀態機器的 Amazon 資源名稱 (ARN) 儲存在文字檔中。您需要提供 ARN，同時提供 Lambda 函數的許可，以啟動新的 Step Functions 數執行。

步驟 4：更新 IAM 政策

為了確保您的 Lambda 函數具有啟動新 Step Functions 數執行的許可，請將內嵌政策附加到您用於 `Restart` Lambda 函數的 IAM 角色。如需詳細資訊，請參閱 IAM 使用者指南中的內嵌內嵌政策。

> **Note**
>
> 您可以更新前一個範例中的 `Resource` 行，以參考 `ContinueAsNew` 狀態機器的 ARN。這將限制政策，使該政策只能啟動特定狀態機器的執行。

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
     }
 ]
}
```

步驟 5：運行狀態機

若要啟動執行，請提供包含狀態機器 ARN 的輸入和 `executionCount`，以指示應啟動新執行的次數。

1. 在ContinueAsNew頁面上，選擇 [開始執行]。

   此時會顯示「開始執行」對話方塊。

2. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   b. 在 [輸入] 方塊中，輸入下列 JSON 輸入以執行您的工作流程。

      ```
      {
        "restart": {
          "StateMachineArn": "arn:aws:states:us-
      east-1:123456789012:stateMachine:ContinueAsNew",
          "executionCount": 4
        }
      }
      ```

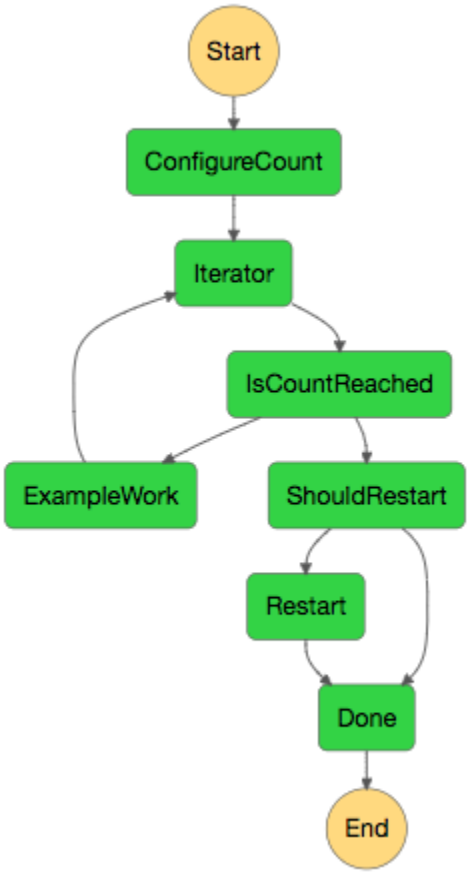
   c. 使用 `ContinueAsNew` 狀態機器的 ARN 來更新 `StateMachineArn` 欄位。

   d. 選擇 Start execution (開始執行)。

   e. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

      「圖形」檢視會顯示四個執行項目中的第一個。在執行完成之前，它將通過 `Restart` 狀態並啟動新的執行。



當執行完成時，您可以查看正在執行的下一個執行項目。選擇頂部的ContinueAsNew鏈接以查看執行列表。您應該會看到最近關閉的執行，以及 `Restart` Lambda 函數啟動的持續執行。

所有執行完成後，您應該會在清單中看到四個成功的執行。第一個開始執行會顯示您選擇的名稱，後續的執行具有產生的名稱。

部署人工核准專案範例

本教學課程會示範如何部署人工核准專案，即讓 AWS Step Functions 執行在任務期間暫停，並等候直到使用者回應電子郵件。當使用者核准該任務可繼續進行時，工作流程就會前往下一個狀態。

部署本教學課程中包含的 AWS CloudFormation 堆疊將會建立所有必要的資源，包括：

- Amazon API Gateway 資源
- 一個 AWS Lambda 函數
- 一个 AWS Step Functions 状态机
- Amazon 簡易通知服務電子郵件主題
- 相關 AWS Identity and Access Management 角色和權限

> **Note**
>
> 建立 AWS CloudFormation 堆疊時，您必須提供有效的電子郵件地址。

若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudFormation 使用 CloudFormation 範本和*AWS::StepFunctions::StateMachine*資源〉。

主題

- 步驟 4：運行狀態機
- AWS CloudFormation 模板源代碼

步驟 1：建立 AWS CloudFormation 範本

1. 複製AWS CloudFormation 模板源代碼一節中的範例程式碼。

2. 將 AWS CloudFormation 範本的來源貼到本機電腦上的檔案中。

   在此範例中，檔案名為 `human-approval.yaml`。

步驟 2：建立堆疊

1. 登入 AWS CloudFormation 主控台。
2. 選擇 [建立堆疊]，然後選擇 [使用新資源 (標準)]。
3. 在 Create stack (建立堆疊) 頁面上，執行下列動作：

   a. 在 [先決條件-準備範本] 區段中，確定已選取 [範本已準備就緒]。

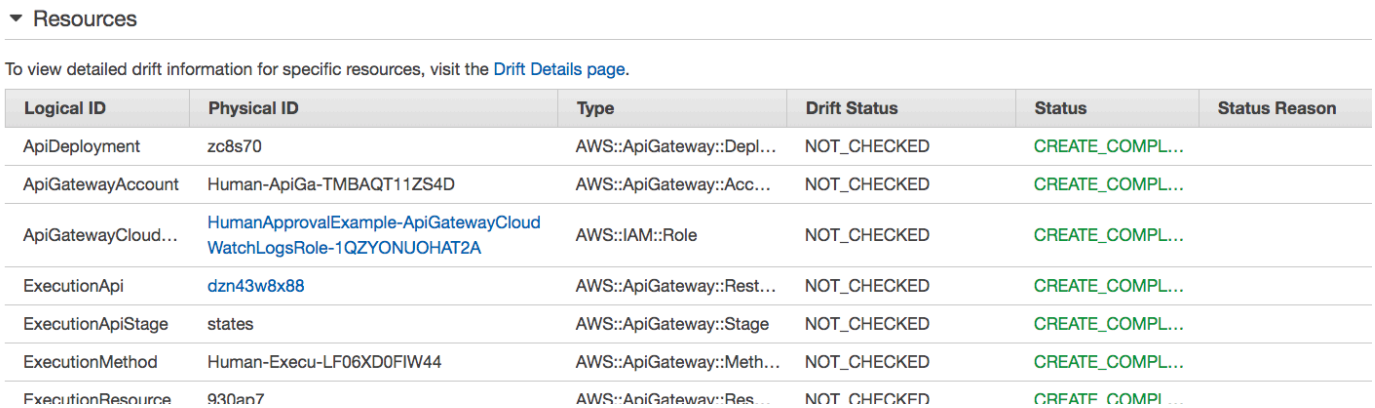
   b. 在 [指定範本] 區段中，選擇 [上傳範本檔案]，然後選擇 [選擇檔案] 以上傳您先前建立的包含範本原始程式碼的`human-approval.yaml`檔案。
4. 選擇下一步。
5. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，執行下列操作：

   a. 在堆疊名稱中，輸入堆疊的名稱。

   b. 在「參數」下，輸入有效的電子郵件地址。您將使用此電子郵件地址訂閱 Amazon SNS 主題。
6. 選擇 [下一步]，然後再選擇 [下一步]。
7. 在 [檢閱] 頁面上，選擇 [我確認 AWS CloudFormation 可能會建立 IAM 資源]，然後選擇 [建立]。

   AWS CloudFormation 會開始建立您的堆疊，並顯示「建立中 _ 進度」狀態。當程序完成時，AWS CloudFormation 會顯示「建立 _ 完成」狀態。

8. (選用) 若要顯示堆疊中的資源，請選取堆疊，然後選擇 Resources (資源) 標籤。

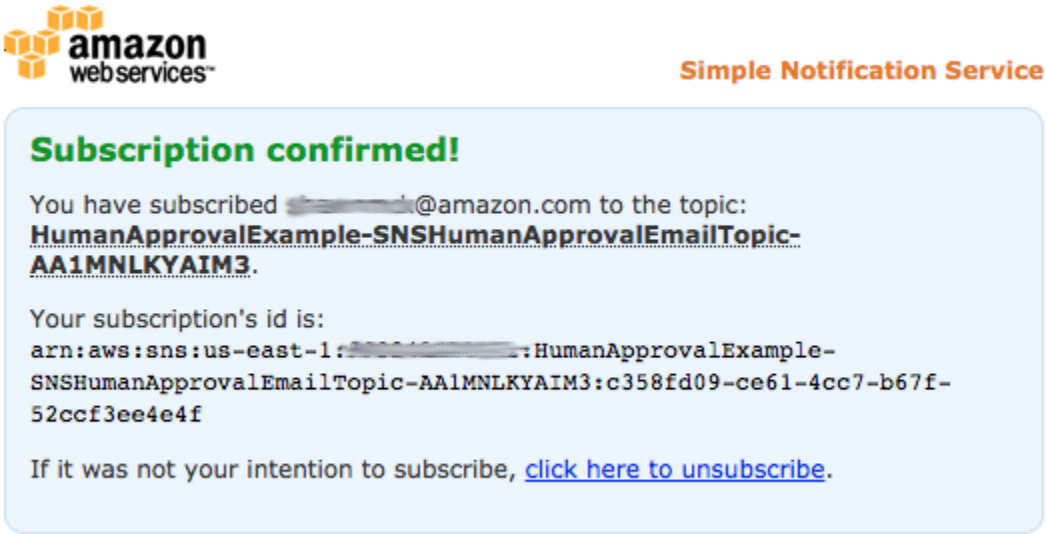


步驟 3：批准 Amazon SNS 訂閱

建立 Amazon SNS 主題後，您會收到一封電子郵件，要求您確認訂閱。

1. 開啟您在建立 AWS CloudFormation 堆疊時提供的電子郵件帳戶。
2. 打開消息AWS 通知-來自 no-reply@sns.amazonaws.com 的訂閱確認

   電子郵件會列出 Amazon SNS 主題的亞馬遜資源名稱，以及一個確認連結。
3. 選擇確認訂閱連結。



步驟 4：運行狀態機

1. 在HumanApprovalLambdaStateMachine頁面上，選擇 [開始執行]。

此時會顯示「開始執行」對話方塊。

2. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   b. 在 [輸入] 方塊中，輸入下列 JSON 輸入以執行您的工作流程。

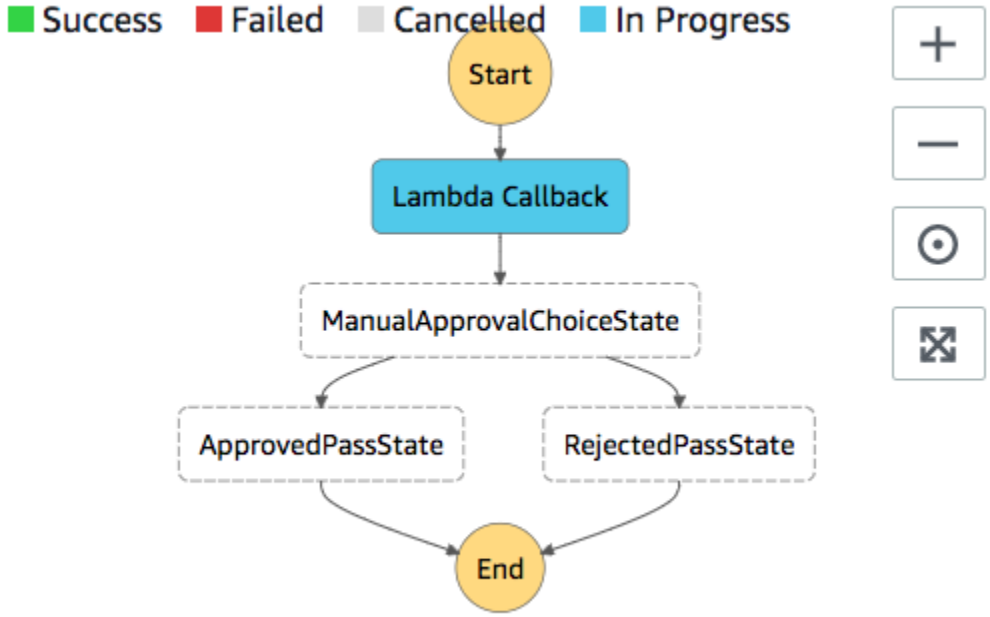
      ```
      {
          "Comment": "Testing the human approval tutorial."
      }
      ```

   c. 選擇 Start execution (開始執行)。

      狀ApprovalTest態機器執行會啟動，並在 Lambda 回呼工作暫停。

   d. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

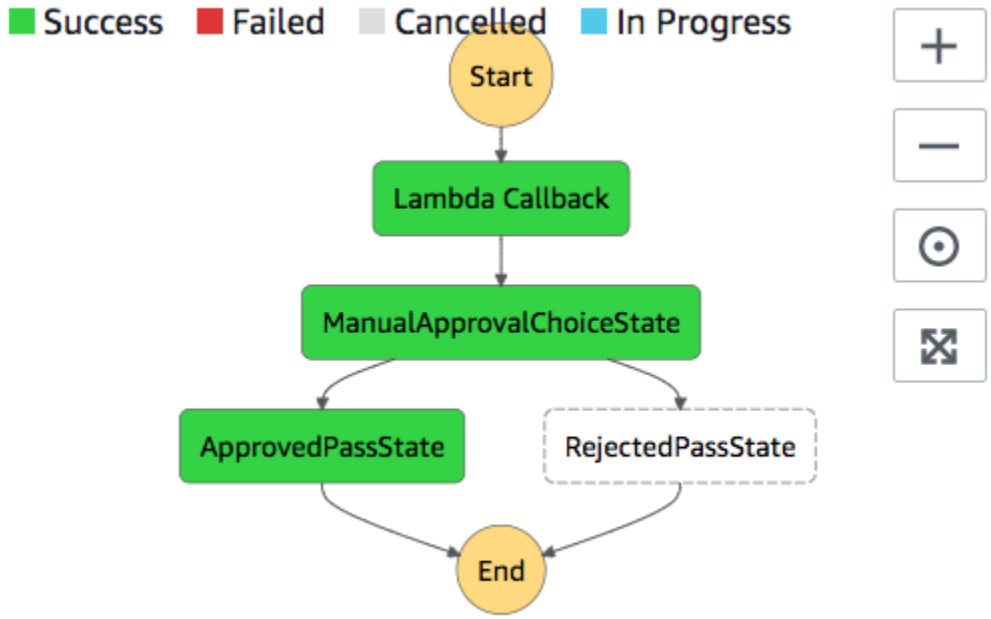


3. 在您之前用於 Amazon SNS 主題的電子郵件帳戶中，開啟主旨為「需要核准來源」的訊息 AWS Step Functions。

   這封訊息會包含分別用於 Approve (核准) 與 Reject (拒絕) 的 URL。

4. 選擇該 Approve (核准) URL。

   工作流程會根據您的選擇繼續進行。



AWS CloudFormation 模板源代碼

使用此 AWS CloudFormation 範本來部署人工核准程序工作流程的範例。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS Step Functions Human based task example. It sends an email with an
 HTTP URL for approval."
Parameters:
  Email:
    Type: String
    AllowedPattern: "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+$"
    ConstraintDescription: Must be a valid email address.
Resources:
  # Begin API Gateway Resources
  ExecutionApi:
    Type: "AWS::ApiGateway::RestApi"
    Properties:
      Name: "Human approval endpoint"
      Description: "HTTP Endpoint backed by API Gateway and Lambda"
      FailOnWarnings: true

  ExecutionResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
```

```
      RestApiId: !Ref ExecutionApi
      ParentId: !GetAtt "ExecutionApi.RootResourceId"
      PathPart: execution

  ExecutionMethod:
    Type: "AWS::ApiGateway::Method"
    Properties:
      AuthorizationType: NONE
      HttpMethod: GET
      Integration:
        Type: AWS
        IntegrationHttpMethod: POST
        Uri: !Sub "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${LambdaApprovalFunction.Arn}/invocations"
        IntegrationResponses:
          - StatusCode: 302
            ResponseParameters:
              method.response.header.Location:
 "integration.response.body.headers.Location"
        RequestTemplates:
          application/json: |
            {
              "body" : $input.json('$'),
              "headers": {
                #foreach($header in $input.params().header.keySet())
                "$header":
 "$util.escapeJavaScript($input.params().header.get($header))"
 #if($foreach.hasNext),#end

                #end
              },
              "method": "$context.httpMethod",
              "params": {
                #foreach($param in $input.params().path.keySet())
                "$param": "$util.escapeJavaScript($input.params().path.get($param))"
 #if($foreach.hasNext),#end

                #end
              },
              "query": {
                #foreach($queryParam in $input.params().querystring.keySet())
                "$queryParam":
 "$util.escapeJavaScript($input.params().querystring.get($queryParam))"
 #if($foreach.hasNext),#end
```

```
              #end
            }
          }
    ResourceId: !Ref ExecutionResource
    RestApiId: !Ref ExecutionApi
    MethodResponses:
      - StatusCode: 302
        ResponseParameters:
          method.response.header.Location: true

ApiGatewayAccount:
  Type: 'AWS::ApiGateway::Account'
  Properties:
    CloudWatchRoleArn: !GetAtt "ApiGatewayCloudWatchLogsRole.Arn"

ApiGatewayCloudWatchLogsRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - apigateway.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    Policies:
      - PolicyName: ApiGatewayLogsPolicy
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
                - "logs:*"
              Resource: !Sub "arn:${AWS::Partition}:logs:*:*:*"

ExecutionApiStage:
  DependsOn:
    - ApiGatewayAccount
  Type: 'AWS::ApiGateway::Stage'
  Properties:
    DeploymentId: !Ref ApiDeployment
```

```
      MethodSettings:
        - DataTraceEnabled: true
          HttpMethod: '*'
          LoggingLevel: INFO
          ResourcePath: /*
      RestApiId: !Ref ExecutionApi
      StageName: states

  ApiDeployment:
    Type: "AWS::ApiGateway::Deployment"
    DependsOn:
      - ExecutionMethod
    Properties:
      RestApiId: !Ref ExecutionApi
      StageName: DummyStage
  # End API Gateway Resources

  # Begin
  # Lambda that will be invoked by API Gateway
  LambdaApprovalFunction:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile:
          Fn::Sub: |
            const { SFN: StepFunctions } = require("@aws-sdk/client-sfn");
            var redirectToStepFunctions = function(lambdaArn, statemachineName,
executionName, callback) {
              const lambdaArnTokens = lambdaArn.split(":");
              const partition = lambdaArnTokens[1];
              const region = lambdaArnTokens[3];
              const accountId = lambdaArnTokens[4];

              console.log("partition=" + partition);
              console.log("region=" + region);
              console.log("accountId=" + accountId);

              const executionArn = "arn:" + partition + ":states:" + region + ":" +
accountId + ":execution:" + statemachineName + ":" + executionName;
              console.log("executionArn=" + executionArn);

              const url = "https://console.aws.amazon.com/states/home?region=" + region
+ "#/executions/details/" + executionArn;
              callback(null, {
```

```
                statusCode: 302,
                headers: {
                  Location: url
                }
              });
            };

            exports.handler = (event, context, callback) => {
              console.log('Event= ' + JSON.stringify(event));
              const action = event.query.action;
              const taskToken = event.query.taskToken;
              const statemachineName = event.query.sm;
              const executionName = event.query.ex;

              const stepfunctions = new StepFunctions();

              var message = "";

              if (action === "approve") {
                message = { "Status": "Approved! Task approved by ${Email}" };
              } else if (action === "reject") {
                message = { "Status": "Rejected! Task rejected by ${Email}" };
              } else {
                console.error("Unrecognized action. Expected: approve, reject.");
                callback({"Status": "Failed to process the request. Unrecognized
Action."});
              }

              stepfunctions.sendTaskSuccess({
                output: JSON.stringify(message),
                taskToken: event.query.taskToken
              })
              .then(function(data) {
                redirectToStepFunctions(context.invokedFunctionArn, statemachineName,
executionName, callback);
              }).catch(function(err) {
                console.error(err, err.stack);
                callback(err);
              });
            }
      Description: Lambda function that callback to AWS Step Functions
      FunctionName: LambdaApprovalFunction
      Handler: index.handler
      Role: !GetAtt "LambdaApiGatewayIAMRole.Arn"
```

```
      Runtime: nodejs18.x

  LambdaApiGatewayInvoke:
    Type: "AWS::Lambda::Permission"
    Properties:
      Action: "lambda:InvokeFunction"
      FunctionName: !GetAtt "LambdaApprovalFunction.Arn"
      Principal: "apigateway.amazonaws.com"
      SourceArn: !Sub "arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:
${ExecutionApi}/*"

  LambdaApiGatewayIAMRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Action:
              - "sts:AssumeRole"
            Effect: "Allow"
            Principal:
              Service:
                - "lambda.amazonaws.com"
      Policies:
        - PolicyName: CloudWatchLogsPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "logs:*"
                Resource: !Sub "arn:${AWS::Partition}:logs:*:*:*"
        - PolicyName: StepFunctionsPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "states:SendTaskFailure"
                  - "states:SendTaskSuccess"
                Resource: "*"
  # End Lambda that will be invoked by API Gateway

  # Begin state machine that publishes to Lambda and sends an email with the link for
 approval
  HumanApprovalLambdaStateMachine:
```

```
    Type: AWS::StepFunctions::StateMachine
    Properties:
      RoleArn: !GetAtt LambdaStateMachineExecutionRole.Arn
      DefinitionString:
        Fn::Sub: |
          {
              "StartAt": "Lambda Callback",
              "TimeoutSeconds": 3600,
              "States": {
                  "Lambda Callback": {
                      "Type": "Task",
                      "Resource": "arn:
${AWS::Partition}:states:::lambda:invoke.waitForTaskToken",
                      "Parameters": {
                        "FunctionName": "${LambdaHumanApprovalSendEmailFunction.Arn}",
                        "Payload": {
                          "ExecutionContext.$": "$$",
                          "APIGatewayEndpoint": "https://${ExecutionApi}.execute-api.
${AWS::Region}.amazonaws.com/states"
                        }
                      },
                      "Next": "ManualApprovalChoiceState"
                  },
                  "ManualApprovalChoiceState": {
                    "Type": "Choice",
                    "Choices": [
                      {
                        "Variable": "$.Status",
                        "StringEquals": "Approved! Task approved by ${Email}",
                        "Next": "ApprovedPassState"
                      },
                      {
                        "Variable": "$.Status",
                        "StringEquals": "Rejected! Task rejected by ${Email}",
                        "Next": "RejectedPassState"
                      }
                    ]
                  },
                  "ApprovedPassState": {
                    "Type": "Pass",
                    "End": true
                  },
                  "RejectedPassState": {
                    "Type": "Pass",
```

```
                  "End": true
                }
            }
        }

SNSHumanApprovalEmailTopic:
  Type: AWS::SNS::Topic
  Properties:
    Subscription:
      -
         Endpoint: !Sub ${Email}
         Protocol: email

LambdaHumanApprovalSendEmailFunction:
  Type: "AWS::Lambda::Function"
  Properties:
    Handler: "index.lambda_handler"
    Role: !GetAtt LambdaSendEmailExecutionRole.Arn
    Runtime: "nodejs18.x"
    Timeout: "25"
    Code:
      ZipFile:
        Fn::Sub: |
          console.log('Loading function');
          const { SNS } = require("@aws-sdk/client-sns");
          exports.lambda_handler = (event, context, callback) => {
              console.log('event= ' + JSON.stringify(event));
              console.log('context= ' + JSON.stringify(context));

              const executionContext = event.ExecutionContext;
              console.log('executionContext= ' + executionContext);

              const executionName = executionContext.Execution.Name;
              console.log('executionName= ' + executionName);

              const statemachineName = executionContext.StateMachine.Name;
              console.log('statemachineName= ' + statemachineName);

              const taskToken = executionContext.Task.Token;
              console.log('taskToken= ' + taskToken);

              const apigwEndpint = event.APIGatewayEndpoint;
              console.log('apigwEndpint = ' + apigwEndpint)
```

```
                const approveEndpoint = apigwEndpint + "/execution?
action=approve&ex=" + executionName + "&sm=" + statemachineName + "&taskToken=" +
 encodeURIComponent(taskToken);
                console.log('approveEndpoint= ' + approveEndpoint);

                const rejectEndpoint = apigwEndpint + "/execution?
action=reject&ex=" + executionName + "&sm=" + statemachineName + "&taskToken=" +
 encodeURIComponent(taskToken);
                console.log('rejectEndpoint= ' + rejectEndpoint);

                const emailSnsTopic = "${SNSHumanApprovalEmailTopic}";
                console.log('emailSnsTopic= ' + emailSnsTopic);

                var emailMessage = 'Welcome! \n\n';
                emailMessage += 'This is an email requiring an approval for a step
 functions execution. \n\n'
                emailMessage += 'Please check the following information and click
 "Approve" link if you want to approve. \n\n'
                emailMessage += 'Execution Name -> ' + executionName + '\n\n'
                emailMessage += 'Approve ' + approveEndpoint + '\n\n'
                emailMessage += 'Reject ' + rejectEndpoint + '\n\n'
                emailMessage += 'Thanks for using Step functions!'

                const sns = new SNS();
                var params = {
                  Message: emailMessage,
                  Subject: "Required approval from AWS Step Functions",
                  TopicArn: emailSnsTopic
                };

                sns.publish(params)
                  .then(function(data) {
                    console.log("MessageID is " + data.MessageId);
                    callback(null);
                  }).catch(
                    function(err) {
                    console.error(err, err.stack);
                    callback(err);
                  });
            }

  LambdaStateMachineExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
```

```
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service: states.amazonaws.com
            Action: "sts:AssumeRole"
      Policies:
        - PolicyName: InvokeCallbackLambda
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "lambda:InvokeFunction"
                Resource:
                  - !Sub "${LambdaHumanApprovalSendEmailFunction.Arn}"

  LambdaSendEmailExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: "sts:AssumeRole"
      Policies:
        - PolicyName: CloudWatchLogsPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "logs:CreateLogGroup"
                  - "logs:CreateLogStream"
                  - "logs:PutLogEvents"
                Resource: !Sub "arn:${AWS::Partition}:logs:*:*:*"
        - PolicyName: SNSSendEmailPolicy
          PolicyDocument:
            Statement:
              - Effect: Allow
                Action:
                  - "SNS:Publish"
                Resource:
```

```
                  - !Sub "${SNSHumanApprovalEmailTopic}"

# End state machine that publishes to Lambda and sends an email with the link for
 approval
Outputs:
  ApiGatewayInvokeURL:
    Value: !Sub "https://${ExecutionApi}.execute-api.${AWS::Region}.amazonaws.com/
states"
  StateMachineHumanApprovalArn:
    Value: !Ref HumanApprovalLambdaStateMachine
```

在 Step Functions 中檢視 X-Ray 軌跡

在本教學課程中，您將學習如何使用 X-Ray 追蹤執行狀態機時發生的錯誤。您可以用AWS X-Ray來視覺化狀態機器的元件、識別效能瓶頸，以及疑難排解導致錯誤的要求。在本教學課程中，您將建立數個隨機產生錯誤的 Lambda 函數，然後您可以使用 X-Ray 追蹤和分析這些函數。

本建立使用 Lambda 的 Step Functions 狀態機器教學課程會引導您建立可呼叫 Lambda 函數的狀態機器。如果您已完成該教學課程，請跳至步驟 2 並使用先前建立的 AWS Identity and Access Management (IAM) 角色。

主題

- 步驟 1：為 Lambda 建立 IAM 角色
- 步驟 2：建立 Lambda 函數
- 步驟 3：再建立兩個 Lambda 函數
- 步驟 4：建立狀態機
- 步驟 5：運行狀態機

步驟 1：為 Lambda 建立 IAM 角色

AWS Lambda 和 AWS Step Functions 可以執行程式碼和存取 AWS 資源 (例如，存放在 Amazon S3 儲存貯體中的資料)。若要維護安全性，您必須授予 Lambda 和 Step Functions 存取這些資源的權限。

Lambda 要求您在建立 Lambda 函數時指派 AWS Identity and Access Management (IAM) 角色，與 Step Functions 數要求您在建立狀態機器時指派 IAM 角色的方式相同。

您可以使用 IAM 主控台建立服務連結角色。

建立角色 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。然後選擇 Create role (建立角色)。
3. 選擇AWS 服務角色類型，然後選擇 Lambda。
4. 選擇 L ambda 使用案例。服務會定義使用案例，以包含服務所需的信任政策。然後選擇 Next: Permissions (下一步：許可)。
5. 選擇一或多個許可政策以連接至角色 (例如 `AWSLambdaBasicExecutionRole`)。請參閱 AWS Lambda 許可模型。

   選取可指派您要角色具有之許可政策旁的方塊，然後選擇 Next: Review (下一步：檢閱)。
6. 輸入 Role name (角色名稱)。
7. (選擇性) 針對 Role description (角色描述)，編輯新服務連結角色的描述。
8. 檢閱角色，然後選擇 Create role (建立角色)。

步驟 2：建立 Lambda 函數

您的 Lambda 函數會隨機擲回錯誤或逾時，產生要在 X-Ray 中檢視的範例資料。

> **Important**
>
> 確保您的 Lambda 函數與狀態機位於相同的 AWS 帳戶和 AWS 區域。

1. 開啟 Lambda 主控台，然後選擇建立函數。
2. 在 Create function (建立函數) 區段中，選擇 Author from scratch (從頭開始撰寫)。
3. 在「基本資訊」區段中，設定 Lambda 函數：
   a. 針對 函數名稱 ，請輸入 `TestFunction1`。
   b. 在執行期選擇 Node.js 18.x。
   c. 針對 Role (角色)，請選擇 Choose an existing role (選擇現有的角色)。
   d. 對於現有角色，請選取您先前建立的 Lambda 角色。

> **Note**
>
> 如果您建立的 IAM 角色未出現在清單中，則該角色可能仍需要幾分鐘才能傳播到 Lambda。

e. 選擇建立函數。

建立 Lambda 函數時，請注意頁面右上角的 Amazon 資源名稱 (ARN)。例如：

```
arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
```

4. 將 Lambda 函數的下列程式碼複製到 ***TestFunction1*** 頁面的「函數」程式碼區段中。

```
function getRandomSeconds(max) {
    return Math.floor(Math.random() * Math.floor(max)) * 1000;
}
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
export const handler = async (event) => {
    if(getRandomSeconds(4) === 0) {
        throw new Error("Something went wrong!");
    }
    let wait_time = getRandomSeconds(5);
    await sleep(wait_time);
    return { 'response': true }
};
```

此代碼創建隨機計時故障，這將用於在狀態機中生成可以使用 X-Ray 跟踪進行查看和分析的示例錯誤。

5. 選擇儲存。

步驟 3：再建立兩個 Lambda 函數

再建立兩個 Lambda 函數。

1. 重複步驟 2 以建立另外兩個 Lambda 函數。對於下一個函數，在函數名稱中，輸入`TestFunction2`。對於最後一個函數，在函數名稱中，輸入`TestFunction3`。

2. 在 Lambda 主控台中，檢查您現在有三個 Lambda 函數`TestFunction1`、`TestFunction2`、和`TestFunction3`。

步驟 4：建立狀態機

在此步驟中，您將使用 Step Functions 主控台建立具有三`Task`種狀態的狀態機器。每個`Task`狀態都會參考三個 Lambda 函數中的一個。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。

> **Important**
>
> 請確定您的狀態機與您先前在步驟 2 和步驟 3 中建立的 Lambda 函數位於相同的 AWS 帳戶和區域。

2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 在本教學課程中，您將在程式碼編輯器. Amazon States Language 若要這麼做，請選擇 [程式碼]。
5. 刪除現有的樣板代碼並粘貼以下代碼。在「工作」狀態定義中，請記得將範例 ARN 取代為您建立的 Lambda 函數的 ARN。

```
{
  "StartAt": "CallTestFunction1",
  "States": {
    "CallTestFunction1": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:test-function1",
      "Catch": [
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "AfterTaskFailed"
        }
      ],
      "Next": "CallTestFunction2"
    },
    "CallTestFunction2": {
```

```
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:test-function2",
      "Catch": [
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "AfterTaskFailed"
        }
      ],
      "Next": "CallTestFunction3"
    },
    "CallTestFunction3": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:test-function3",
      "TimeoutSeconds": 5,
      "Catch": [
        {
          "ErrorEquals": [
            "States.Timeout"
          ],
          "Next": "AfterTimeout"
        },
        {
          "ErrorEquals": [
            "States.TaskFailed"
          ],
          "Next": "AfterTaskFailed"
        }
      ],
      "Next": "Succeed"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "AfterTimeout": {
      "Type": "Fail"
    },
    "AfterTaskFailed": {
      "Type": "Fail"
    }
  }
}
```

這是您使用 Amazon 州語言的狀態機器的描述。它定義了三個Task狀態命名為CallTestFunction1，CallTestFunction2和CallTestFunction3。每個函數都會呼叫三個 Lambda 函數之一。如需詳細資訊，請參閱狀態機器結構。

6. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **`TraceFunctions`**。

7. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   在此自學課程的其他組態下，選擇啟用 X-Ray 追蹤。保留狀態機器設定中的所有其他預設選項。

   如果您先前已使用狀態機器的正確許可建立 IAM 角色，並且想要使用它，請在 [權限] 中選取 [選擇現有角色]，然後從清單中選取角色。或選取 [輸入角色 ARN]，然後為該 IAM 角色提供 ARN。

8. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

> **Note**
>
> 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

步驟 5：運行狀態機

狀態機器執行是執行工作流程以執行工作的執行個體。

1. 在***`TraceFunctions`***頁面上，選擇 [開始執行]。

   系統會隨即顯示 New execution (新執行) 頁面。

2. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

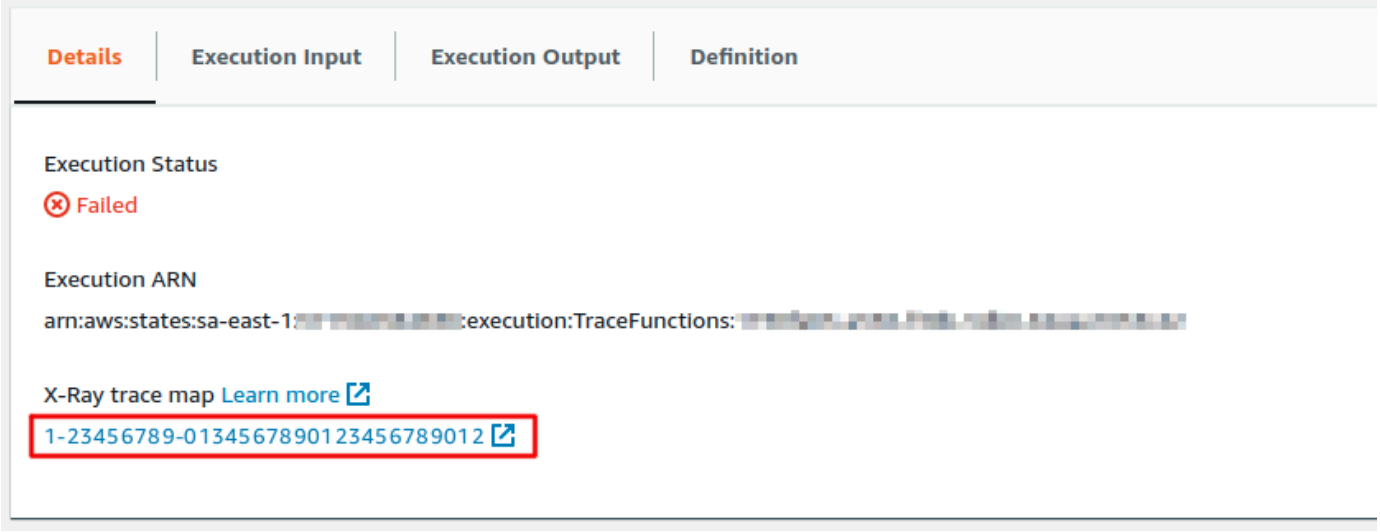
b. 選擇 Start execution (開始執行)。

c. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

運行幾個（至少三個）執行。

3. 執行完成後，請按照 X-Ray 跟踪圖鏈接進行操作。您可以在執行仍在執行時檢視追蹤，但您可能希望在檢視 X-Ray 追蹤對應之前查看執行結果。



4. 檢視服務對應以識別發生錯誤的位置、高延遲的連線，或追蹤失敗的要求。在此示例中，您可以查看每個功能接收的流量。 `TestFunction2`被稱為更頻繁`TestFunction3`，並且`TestFunction1`被稱為兩倍以上的頻率`TestFunction2`。

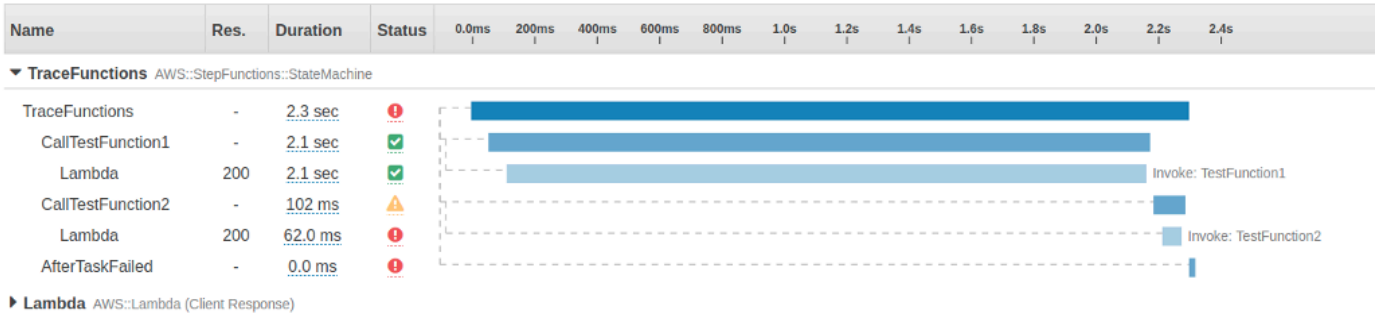
服務映射會透過根據成功呼叫與錯誤及故障的比例，將每個節點標上顏色，來指出每個節點的運作狀態。

- 綠色表示成功呼叫

- 紅色表示伺服器故障 (500 系列錯誤)
- 黃色表示用戶端錯誤 (400 系列錯誤)
- 紫色表示調節錯誤 (429 請求數太多)

您也可以選擇服務節點來檢視該節點的要求，或選擇兩個節點之間的邊緣來檢視傳遞該連線的要求。

5. 檢視 X-Ray 追蹤對應，以查看每次執行的情況。Timeline (時間表) 檢視會顯示區段和子區段的階層。清單中的第一個項目是區段，代表服務為單一請求記錄的所有資料。在區段下方為子區段。此範例顯示 Lambda 函數記錄的子區段。



如需瞭解 X-Ray 追蹤以及使用 X-Ray 與 Step Functions 的詳細資訊，請參閱 AWS X-Ray 和 Step Functions

使用 AWS SDK 服務整合收集 Amazon S3 儲存貯體資訊

本教學課程說明如何使用 Amazon 簡單儲存服務執行 AWS SDK 整合。您在本教學中建立的狀態機會收集 Amazon S3 儲存貯體的相關資訊，然後列出儲存貯體以及目前區域中每個儲存貯體的版本資訊。

主題

步驟 1：建立狀態機

使用 Step Functions 主控台，您將建立一個狀態機器，其中包含列出目前帳戶和區域中所有 Amazon S3 儲存貯體的`Task`狀態。然後，您將添加另一個調用 `HeadBucket` API 的`Task`狀態，以驗證返回的存儲桶是否可以在當前區域中訪問。如果存儲桶無法訪問，則 `HeadBucket` API 調用返回`S3.S3Exception`錯誤。您將包含一個`Catch`塊來 catch 此異常和一個`Pass`狀態作為後備狀態。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在「選擇範本」對話方塊中，選取「空白」。
3. 選擇選取。這會在設計模式中開啟工作流程工作室
4. 在本教學課程中，您將在程式碼編輯器. Amazon States Language 若要這麼做，請選擇 [程式碼]。
5. 刪除現有的樣板代碼並粘貼以下狀態機定義。

```
{
  "Comment": "A description of my state machine",
  "StartAt": "ListBuckets",
  "States": {
    "ListBuckets": {
      "Type": "Task",
      "Parameters": {},
      "Resource": "arn:aws:states:::aws-sdk:s3:listBuckets",
      "Next": "Map"
    },
    "Map": {
      "Type": "Map",
```

```
      "ItemsPath": "$.Buckets",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "INLINE"
        },
        "StartAt": "HeadBucket",
        "States": {
          "HeadBucket": {
            "Type": "Task",
            "ResultPath": null,
            "Parameters": {
              "Bucket.$": "$.Name"
            },
            "Resource": "arn:aws:states:::aws-sdk:s3:headBucket",
            "Catch": [
              {
                "ErrorEquals": [
                  "S3.S3Exception"
                ],
                "ResultPath": null,
                "Next": "Pass"
              }
            ],
            "Next": "GetBucketVersioning"
          },
          "GetBucketVersioning": {
            "Type": "Task",
            "End": true,
            "Parameters": {
              "Bucket.$": "$.Name"
            },
            "ResultPath": "$.BucketVersioningInfo",
            "Resource": "arn:aws:states:::aws-sdk:s3:getBucketVersioning"
          },
          "Pass": {
            "Type": "Pass",
            "End": true,
            "Result": {
              "Status": "Unknown"
            },
            "ResultPath": "$.BucketVersioningInfo"
          }
        }
      },
```

```
      "End": true
    }
  }
}
```

6. 指定狀態機的名稱。若要執行此操作，請選擇的預設狀態機器名稱旁邊的編輯圖示MyStateMachine。然後，在 [狀態機器組態] 中，在 [狀態機器名稱] 方塊中指定名稱。

   針對本教學課程，輸入名稱 **`Gather-S3-Bucket-Info-Standard`**。

7. (選擇性) 在狀態機器組態中，指定其他工作流程設定，例如狀態機器類型及其執行角色。

   保留狀態機器設定中的所有預設選項。

   如果您先前已使用狀態機器的正確許可建立 IAM 角色，並且想要使用它，請在 [權限] 中選取 [選擇現有角色]，然後從清單中選取角色。或選取 [輸入角色 ARN]，然後為該 IAM 角色提供 ARN。

8. 在 [確認角色建立] 對話方塊中，選擇 [確認] 以繼續。

   您也可以選擇 [檢視角色設定] 以返回 [狀態機器組態]。

   > **Note**
   >
   > 如果您刪除 Step Functions 建立的 IAM 角色，則 Step Functions 稍後無法重新建立。同樣地，如果您修改角色 (例如，透過從 IAM 政策中的主體移除 Step Functions)，Step Functions 稍後無法還原其原始設定。

   在步驟 2 中，您將缺少的權限添加到狀態機器角色。

步驟 2：新增必要的 IAM 角色許可

若要收集目前區域中 Amazon S3 儲存貯體的相關資訊，您必須提供狀態機器存取 Amazon S3 儲存貯體的必要許可。

1. 在狀態機器頁面上，選擇 IAM 角色 ARN 以開啟狀態機器角色的「角色」頁面。
2. 選擇新增許可，然後選擇建立內嵌政策。
3. 選擇 [JSON] 索引標籤，然後將下列權限貼到 JSON 編輯器中。

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

```
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
}
```

4. 選擇檢閱政策。
5. 在 檢閱政策 下，針對政策 名稱 ，輸入 **s3-bucket-permissions**。
6. 選擇 建立政策 。

步驟 3：執行標準狀態機器執行

1. 在 [聚集-S3 儲存區資訊標準] 頁面上，選擇 [開始執行]。
2. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   b. 選擇 Start execution (開始執行)。

   c. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

步驟 4：執行快速狀態機器執行

1. 使用步驟 1 中提供的狀態機定義建立 Express 狀態機。請確定您也包含必要的 IAM 角色許可，如步驟 2 所述。

   > **Tip**
   >
   > 若要與之前建立的標準機器區分開來，請將 Express 狀態機命名為**`Gather-S3-Bucket-Info-Express`**。

2. 在 [聚集-S3 儲存區資訊標準] 頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   b. 選擇 Start execution (開始執行)。
   c. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

開發人員工具

以下資源包含有關建置無伺服器工作流程和使用狀態機器的其他資訊：

- AWS CDK
- AWS Toolkit for VS Code

以下主題包含的資訊可教導您如何建立、測試和偵錯狀態機器。

主題

- 開發選項
- AWS Step Functions 而且 AWS SAM
- 使用工作流程工作室 Application Composer
- 使用建立 Step Functions 的 Lambda 狀態機 AWS CloudFormation
- 建立要Step Functions使用的Lambda狀態機 AWS CDK
- 使用同步快速狀態機器建立 API Gateway REST API AWS CDK
- AWS Step Functions 數據科學開發套件
- 使用地形部署狀態機

開發選項

您可以透過多種方式實作 AWS Step Functions 狀態機器，例如使用主控台、SDK 或本機版本的 Step Functions 進行測試和開發。

主題

- Step Functions 控制台
- AWS 開發套件
- 標準和快速工作流程
- HTTPS 服務介面
- 開發環境
- 端點

- AWS CLI
- Step Functions 本地
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model 和 Step Functions
- 地形和 Step Functions
- 定義格式支援

Step Functions 控制台

您可以使用 Step Functions 主控台定義狀態機器。您可以使用為工作提供程式碼，在不使用本機開發環境的情況下，在雲端中撰寫複雜的狀態機器。 AWS Lambda 撰寫完成之後，您就可以使用 Step Functions 主控台使用 Amazon 州語言定義狀態機器。

建立 Lambda 狀態機器教學課程會使用此技巧來建立簡單的狀態機器、執行機器，以及檢視其結果。

數據流模擬器

您可以在 Step Functions 主控台中設計、實作和偵錯工作流程。您也可以使用`JsonPath`輸入和輸出處理來控制工作流程中的資料流程。使用 Step Functions 主控台中的資料流程模擬器，瞭解資訊如何從一個狀態流動到另一個狀態，並瞭解如何篩選和操作資料。此工具會模擬 Step Functions 數用來處理資料的每個欄位，例如`InputPathParameters`、`ResultSelector`、`OutputPath`、和`ResultPath`。

如需詳細資訊，請參閱 數據流模擬器。

AWS 開發套件

Step Functions 是由 Java，淨，紅寶石，PHP，Python（博托 3） JavaScript，圍棋和 C ++ 的 AWS SDK 支持。這些 SDK 提供了在多種程式設計語言中使用 Step Functions 式 HTTPS API 動作的便利方式。

您可以使用這些軟體開發套件程式庫所公開的 API 動作來開發狀態機器、活動或狀態機器啟動者。您也可以使用這些程式庫存取可見性作業，以開發自己的 Step Functions 監視和報告工具。

若要將 Step Functions 與其他 AWS 服務搭配使用，請參閱目前 AWS 開發套件和 Amazon Web Services 工具的參考文件。

> **Note**
>
> Step Functions 僅支援 HTTPS 端點。

標準和快速工作流程

當您建立新的狀態機時，您必須選取標準或快速**Type**的一個。在這兩種情況下，您都可以使用 Amazon 州語言定義狀態機器。根據您選取的 Type (類型)，狀態機器執行的行為會有所不同。建立狀態機後，您選擇的「類型」無法變更。

如需詳細資訊，請參閱記錄使用CloudWatch日誌。

HTTPS 服務介面

Step Functions 提供可透過 HTTPS 要求存取的服務作業。您可以使用這些作業直接與 Step Functions 式通訊，並以任何可透過 HTTPS 與 Step Functions 式通訊的語言來開發您自己的程式庫。

您可以使用服務 API 動作來開發狀態機器、工作者或狀態機器啟動者。您也可以透過 API 動作來存取可見度操作，以開發專屬的監控和報告工具。

如需 API 動作的詳細資訊，請參閱 AWS Step Functions API 參考資料。

開發環境

您必須設定與您計劃使用之程式設計語言相容的開發環境。

例如，若要使用 Java 開發 Step Functions 式，您必須在每個開發工作站上安裝 Java 開發環境 AWS SDK for Java，例如。如果您使用適用於 Java 開發人員的 Eclipse IDE，您也應該安裝 AWS Toolkit for Eclipse。此 Eclipse 外掛程式會新增適合在 AWS上進行開發的實用功能。

如果您的程式設計語言需要執行階段環境，則必須在每部執行這些程序的電腦上設定環境。

端點

為了減少延遲並將資料儲存在符合您需求的位置，Step Functions 在不同 AWS 區域提供端點。

Step Functions 中的每個端點是完全獨立的。狀態機器或活動只存在於其建立所在的區域內。您在某區域中建立的任何狀態機器和活動，不會與另一個區域中建立的任何狀態機器和活動共用任何資料或屬性。例如，您可以註冊STATES-Flows-1在兩個不同區域中命名的狀態機器。一個區域中的STATES-Flows-1狀態機不會與另一個地區的STATES-Flow-1狀態機共享數據或屬性。

如需 Step Functions 端點的清單，請參閱AWS Step FunctionsAWS 一般參考.

AWS CLI

您可以從 AWS Command Line Interface (AWS CLI) 存取許多 Step Functions。 AWS CLI 這是使用「Step Functions」主控台的替代方法，或在某些情況下，使用 Step Functions API 動作進行程式設計。例如，您可以使用建立狀態機器，然後列出現有的狀態機器。 AWS CLI

您可以使用中的「Step Functions」命令 AWS CLI 來啟動和管理執行、輪詢活動、記錄工作活動訊號等。如需 Step Functions 命令的完整清單、可用引數的描述以及顯示其用法的範例，請參閱《命AWS CLI 令參考》。

AWS CLI 命令密切遵循 Amazon 州語言，因此您可以使用 AWS CLI 來了解 Step Functions API 動作。您也可以使用現有的 API 知識，從命令列建立程式碼的原型或執行 Step Functions 動作。

Step Functions 本地

為了測試和開發目的，您可以在本機電腦上安裝和執行 Step Functions。隨著 Step Functions 局部，你可以在任何機器上開始執行。

Step Functions 的本機版本可以叫用 AWS Lambda 函式，無論是在 AWS 本機執行或執行時。您也可以協調其他支援的 AWS 服務。如需詳細資訊，請參閱 在本地測試狀態機。

> **Note**
>
> Step Functions 本地使用虛擬帳戶工作。

AWS Toolkit for Visual Studio Code

您可以使用 VS Code 與遠端狀態機器互動，並在本機開發狀態機器。您可以建立或更新狀態機、列出現有的狀態機器，以及執行或下載狀態機器。VS Code 還可讓您從範本建立新狀態機器、檢視狀態機器的視覺效果，並提供程式碼片段、程式碼完成和程式碼驗證。

若要取得更多資訊，請參閱使 AWS Toolkit for Visual Studio Code 用者指南

AWS Serverless Application Model 和 Step Functions

Step Functions 與整合 AWS Serverless Application Model，可讓您整合工作流程與 Lambda 函數、API 和事件，以建立無伺服器應用程式。

您也可以將 AWS SAM CLI 與 AWS Toolkit for Visual Studio Code 作為整合體驗的一部分結合使用。

如需更多資訊，請參閱AWS Step Functions 和 AWS SAM。

地形和 Step Functions

Terraform by HashiCorp 是使用基礎架構作為代碼（IaC）構建應用程序的框架。使用 Terraform，您可以建立狀態機器並使用功能，例如預覽基礎結構部署和建立可重複使用的範本。Terraform 範本可協助您維護和重複使用程式碼，方法是將程式碼分解成較小的區塊。

如需詳細資訊，請參閱 使用地形部署狀態機。

定義格式支援

Step Functions 提供了各種工具，可讓您以不同的格式提供狀態機定義。指定狀態機器詳細資料的 Amazon 州語言 (ASL) 定義可以作為字串提供，也可以使用 JSON 或 YAML 作為序列化物件提供。

> **Note**
>
> YAML 允許單行註釋。範本的狀態機器定義部分中提供的任何 YAML 註解都不會結轉至建立的資源定義中。相反地，您可以使用狀態機器定義中的內Comment容。如需詳細資訊，請參閱国家机结构頁面。

下表顯示哪些工具支援以 ASL 為基礎的定義。

工具定義格式支援

	JSON	YAML	字符串化的 Amazon 國家語言		
Step Functions 控制台	✓				
HTTPS 服務接口			✓		
AWS CLI			✓		

	JSON	YAML	字符串化的 Amazon 國家語言		
Step Functions 本地			✓		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation AWS SAM 也可讓您以 JSON 或 YAML 格式將狀態機器定義上傳到 Amazon S3，並在範本中提供定義的 Amazon S3 位置。當您的狀態機定義很複雜時，這可以提高模板的可讀性。如需詳細資訊，請參閱 AWS::StepFunctions::StateMachine S3 位置頁面。

下列 AWS CloudFormation 範例範本顯示如何使用不同的輸入格式提供相同的狀態機器定義。

JSON with Definition

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "AWS Step Functions sample template.",
  "Resources": {
    "MyStateMachine": {
      "Type": "AWS::StepFunctions::StateMachine",
      "Properties": {
        "RoleArn": {
          "Fn::GetAtt": [ "StateMachineRole", "Arn" ]
```

```
        },
        "TracingConfiguration": {
          "Enabled": true
        },
        "Definition": {
          "StartAt": "HelloWorld",
          "States": {
            "HelloWorld": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      }
    },
    "StateMachineRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "states.amazonaws.com"
                ]
              }
            }
          ]
        },
        "ManagedPolicyArns": [],
        "Policies": [
          {
            "PolicyName": "StateMachineRolePolicy",
            "PolicyDocument": {
              "Statement": [
                {
                  "Action": [
                    "lambda:InvokeFunction"
                  ],
```

```
                  "Resource": "*",
                  "Effect": "Allow"
                }
              ]
            }
          }
        ]
      }
    }
  },
  "Outputs": {
    "StateMachineArn": {
      "Value": {
        "Ref": "MyStateMachine"
      }
    }
  }
}
```

JSON with DefinitionString

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "AWS Step Functions sample template.",
  "Resources": {
    "MyStateMachine": {
      "Type": "AWS::StepFunctions::StateMachine",
      "Properties": {
        "RoleArn": {
          "Fn::GetAtt": [ "StateMachineRole", "Arn" ]
        },
        "TracingConfiguration": {
          "Enabled": true
        },
        "DefinitionString": "{\n  \"StartAt\": \"HelloWorld\",\n  \"States\": {\n
 \"HelloWorld\": {\n      \"Type\": \"Pass\",\n      \"End\": true\n    }\n  }\n}"
      }
    },
    "StateMachineRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
```

```
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "states.amazonaws.com"
                ]
              }
            }
          ]
        },
        "ManagedPolicyArns": [],
        "Policies": [
          {
            "PolicyName": "StateMachineRolePolicy",
            "PolicyDocument": {
              "Statement": [
                {
                  "Action": [
                    "lambda:InvokeFunction"
                  ],
                  "Resource": "*",
                  "Effect": "Allow"
                }
              ]
            }
          }
        ]
      }
    }
  },
  "Outputs": {
    "StateMachineArn": {
      "Value": {
        "Ref": "MyStateMachine"
      }
    }
  }
}
```

YAML with Definition

```
AWSTemplateFormatVersion: 2010-09-09
Description: AWS Step Functions sample template.
Resources:
  MyStateMachine:
    Type: 'AWS::StepFunctions::StateMachine'
    Properties:
      RoleArn: !GetAtt
        - StateMachineRole
        - Arn
      TracingConfiguration:
        Enabled: true
      Definition:
        # This is a YAML comment. This will not be preserved in the state machine
 resource's definition.
        Comment: This is an ASL comment. This will be preserved in the state machine
 resource's definition.
        StartAt: HelloWorld
        States:
          HelloWorld:
            Type: Pass
            End: true
  StateMachineRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - states.amazonaws.com
      ManagedPolicyArns: []
      Policies:
        - PolicyName: StateMachineRolePolicy
          PolicyDocument:
            Statement:
              - Action:
                  - 'lambda:InvokeFunction'
                Resource: "*"
                Effect: Allow
```

```
Outputs:
  StateMachineArn:
    Value:
      Ref: MyStateMachine
```

YAML with DefinitionString

```
AWSTemplateFormatVersion: 2010-09-09
Description: AWS Step Functions sample template.
Resources:
  MyStateMachine:
    Type: 'AWS::StepFunctions::StateMachine'
    Properties:
      RoleArn: !GetAtt
        - StateMachineRole
        - Arn
      TracingConfiguration:
        Enabled: true
      DefinitionString: |
        {
            "StartAt": "HelloWorld",
            "States": {
                "HelloWorld": {
                    "Type": "Pass",
                    "End": true
                }
            }
        }
  StateMachineRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - states.amazonaws.com
      ManagedPolicyArns: []
      Policies:
```

```
        - PolicyName: StateMachineRolePolicy
          PolicyDocument:
            Statement:
              - Action:
                  - 'lambda:InvokeFunction'
                Resource: "*"
                Effect: Allow

Outputs:
  StateMachineArn:
    Value:
      Ref: MyStateMachine
```

AWS Step Functions 而且 AWS SAM

您可以將 AWS SAM CLI 與 AWS Toolkit for Visual Studio Code 作為整合體驗的一部分結合使用，在本機建立狀態機器。您可以使用建置無伺服器應用程式 AWS SAM，然後在 VS Code IDE 中建置您的狀態機器。然後，您可以驗證、封裝和部署您的資源。或者，您也可以發佈至 AWS Serverless Application Repository.

> **Tip**
>
> 若要部署啟動 Step Functions 工作流程 AWS SAM的範例無伺服器應 AWS SAM 用程式 AWS 帳戶，請參閱 AWS Step Functions 工作坊的單元 11-部署。

主題

為什麼要搭配使用 Step Functions AWS SAM？

當您搭配使用 Step Functions 時， AWS SAM 您可以：

- 開始使用範 AWS SAM 例範本。
- 將您的狀態機器建置到無服務器應用程式中。
- 使用變數替代，在部署時將 ARN 替換為狀態機器。

  AWS CloudFormation支援 `DefinitionSubstitutions`，可讓您將工作流程定義中的動態參照新增至您在CloudFormation範本中提供的值。您可以使用`${dollar_sign_brace}`符號將替代項新增至工作流程定義，藉此新增動態參照。您還需要在CloudFormation模板中的 StateMachine 資源的`DefinitionSubstitutions`屬性中定義這些動態引用。在CloudFormation堆棧創建過程中，這些替換被實際值替換。如需詳細資訊，請參閱 DefinitionSubstitutions 在AWS SAM範本中。
- 使用 AWS SAM 原則範本指定狀態機的角色。
- 使用 API Gateway、 EventBridge 事件或 AWS SAM 範本中的排程啟動狀態機器執行。

Step Functions 與 AWS SAM 規範整合

您可以使用AWS SAM 原則範本將權限新增至狀態機器。透過這些權限，您可以協調 Lambda 函數和其他 AWS 資源，以形成複雜且強大的工作流程。

Step Functions 與 SAM CLI 整合

Step Functions 與 AWS SAM CLI 集成。使用此功能可將狀態機快速開發為無伺服器應用程式。

請嘗試使用創建 Step Functions 狀態機AWS SAM教學課程，瞭解如 AWS SAM 何使用建立狀態機器。

支援的 AWS SAM CLI 功能包括：

CLI 命令	描述
sam init	使用範本初始化無伺服器應用程式。 AWS SAM 可與 Step Functions 的 SAM 範本搭配使用。
sam validate	驗證 AWS SAM 範本。
sam package	封裝 AWS SAM 應用程式。它會建立您的程式碼和相依性的 ZIP 檔案，然後將其上傳到 Amazon S3。然後它會傳回 AWS SAM 範本的副本，以命令上傳成品的 Amazon S3 位置取代本機成品的參考。

CLI 命令	描述
sam deploy	部署 AWS SAM 應用程式。
sam publish	將 AWS SAM 應用程式發佈到 AWS Serverless Application Repository. 此指令會取得封裝 AWS SAM 範本，並將應用程式發佈至指定的區域。

> **Note**
>
> 使用 AWS SAM 本機時，您可以在本機模擬 Lambda 和 API Gateway。但是，您無法在本地使用 AWS SAM.

DefinitionSubstitutions 在AWS SAM範本中

您可以使用CloudFormation範本來定義狀態機器AWS SAM。使用 AWS SAM，您可以定義內嵌在範本或單獨的檔案中的狀態機器。以下AWS SAM模板包括模擬股票交易工作流程的狀態機。該狀態機調用三個Lambda功能來檢查股票的價格，並確定是買入還是賣出股票。然後將此交易記錄在Amazon DynamoDB表格中。下列範本中Lambda函數和DynamoDB表格的 ARN 是使用DefinitionSubstitutions指定的。

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: |
  step-functions-stock-trader
  Sample SAM Template for step-functions-stock-trader
Resources:
  StockTradingStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionSubstitutions:
        StockCheckerFunctionArn: !GetAtt StockCheckerFunction.Arn
        StockSellerFunctionArn: !GetAtt StockSellerFunction.Arn
        StockBuyerFunctionArn: !GetAtt StockBuyerFunction.Arn
        DDBPutItem: !Sub arn:${AWS::Partition}:states:::dynamodb:putItem
        DDBTable: !Ref TransactionTable
      Policies:
        - DynamoDBWritePolicy:
```

```
            TableName: !Ref TransactionTable
        - LambdaInvokePolicy:
            FunctionName: !Ref StockCheckerFunction
        - LambdaInvokePolicy:
            FunctionName: !Ref StockBuyerFunction
        - LambdaInvokePolicy:
            FunctionName: !Ref StockSellerFunction
      DefinitionUri: statemachine/stock_trader.asl.json
  StockCheckerFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: functions/stock-checker/
      Handler: app.lambdaHandler
      Runtime: nodejs18.x
      Architectures:
        - x86_64
  StockSellerFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: functions/stock-seller/
      Handler: app.lambdaHandler
      Runtime: nodejs18.x
      Architectures:
        - x86_64
  StockBuyerFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: functions/stock-buyer/
      Handler: app.lambdaHandler
      Runtime: nodejs18.x
      Architectures:
        - x86_64
  TransactionTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: S
```

下面的代碼是在該使用創建 Step Functions 狀態機AWS SAM教程中使用的文件中stock_trader.asl.json的狀態機定義。這個狀態機定義包含幾個DefinitionSubstitutions由符號表示。${dollar_sign_brace}例如，而不是為Check Stock Value任務指定靜態Lambda函數 ARN，而${StockCheckerFunctionArn}是使用替代。

此替代是在模板的DefinitionSubstitutions屬性中定義的。 DefinitionSubstitutions是狀態機器資源的鍵值對映。在中DefinitionSubstitutions，$ {StockCheckerFunctionArn} 映射到使用 CloudFormation內在函數StockCheckerFunction資源的 ARN。!GetAtt當您部署AWS SAM範本時，範本DefinitionSubstitutions中的會被實際值取代。

```
{
    "Comment": "A state machine that does mock stock trading.",
    "StartAt": "Check Stock Value",
    "States": {
        "Check Stock Value": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
                "Payload.$": "$",
                "FunctionName": "${StockCheckerFunctionArn}"
            },
            "Next": "Buy or Sell?"
        },
        "Buy or Sell?": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.stock_price",
                    "NumericLessThanEquals": 50,
                    "Next": "Buy Stock"
                }
            ],
            "Default": "Sell Stock"
        },
        "Buy Stock": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
                "Payload.$": "$",
                "FunctionName": "${StockBuyerFunctionArn}"
            },
            "Retry": [
                {
                    "ErrorEquals": [
                        "Lambda.ServiceException",
                        "Lambda.AWSLambdaException",
```

```
                    "Lambda.SdkClientException",
                    "Lambda.TooManyRequestsException"
                ],
                "IntervalSeconds": 1,
                "MaxAttempts": 3,
                "BackoffRate": 2
            }
        ],
        "Next": "Record Transaction"
    },
    "Sell Stock": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "OutputPath": "$.Payload",
        "Parameters": {
            "Payload.$": "$",
            "FunctionName": "${StockSellerFunctionArn}"
        },
        "Next": "Record Transaction"
    },
    "Record Transaction": {
        "Type": "Task",
        "Resource": "arn:aws:states:::dynamodb:putItem",
        "Parameters": {
            "TableName": "${DDBTable}",
            "Item": {
                "Id": {
                    "S.$": "$.id"
                },
                "Type": {
                    "S.$": "$.type"
                },
                "Price": {
                    "N.$": "$.price"
                },
                "Quantity": {
                    "N.$": "$.qty"
                },
                "Timestamp": {
                    "S.$": "$.timestamp"
                }
            }
        },
        "End": true
```

```
        }
    }
}
```

後續步驟

您可以透過下列資源進一步瞭解如何使用步驟函數： AWS SAM

- 完成使用創建 Step Functions 狀態機AWS SAM教學課程以使用建立狀態機器 AWS SAM。
- 指定AWS::Serverless::StateMachine資源。
- 尋找要使用的 AWS SAM 政策範本。
- AWS Toolkit for Visual Studio Code與 Step Functions 搭配使用。
- 檢閱 AWS SAM CLI 參考，以進一步瞭解中可用功能 AWS SAM。

您也可以設計和建置您的工作流程中的基礎設施作為程式碼 (IaC) 使用視覺化建置工作流程, 如. Application Composer 如需詳細資訊，請參閱 使用工作流程工作室 Application Composer。

使用工作流程工作室 Application Composer

AWS 應用程式編寫器是一個可視化生成器，可幫助您使用簡單的圖形界面開發AWS SAM和AWS CloudFormation模板。使用Application Composer，您可以透過AWS 服務在視覺畫布中拖曳、分組和連線來設計應用程式架構。 Application Composer然後從您的設計中創建一個基礎結構作為代碼（IaC）模板，您可以使用該模板來部署應用程序使用AWS SAM命令行界面（AWS SAMCLI）或CloudFormation. 若要進一步了解 Application Composer，請參閱什麼是 Application Composer。

工作流程 Studio 可在中協Application Composer助您設計和建置工作流程。工作流 Studio 中Application Composer提供了一個可視化的 IaC 環境，使您可以輕鬆地將工作流程納入使用 IaC 工具，如模板構建的無服務器應用程序。CloudFormation當您在中使用 Workflow Studio 時Application Composer，它會將個別工作流程步驟連接至AWS資源，並在AWS SAM範本中產生資源組態。它也會新增工作流程執行所需的IAM權限。使用中的工作流程 StudioApplication Composer，您可以建立應用程式的原型，並將它們轉換為生產就緒應用程式。

當您在中使用工作流程工作室時Application Composer，您可以在Application Composer畫布和工作流程工作室之間來回切換。

主題

- 使用工作流程 Studio 建Application Composer置無伺服器工作流程
- 使用工作流程 Studio 中的CloudFormation定義替代動態參考資源
- 將服務整合工作 Connect 至增強型元件卡
- 導入現有項目並在本地同步
- 不可用的工作流程 Studio 功能 AWS 應用程式編寫器

使用工作流程 Studio 建Application Composer置無伺服器工作流程

1. 開啟應用程式撰寫器主控台，然後選擇 [建立專案] 以建立專案。
2. 在資源面板的搜尋欄位中，輸入 **`state machine`**。
3. 將狀Step Functions態機器資源拖到畫布上。
4. 在工作流程 Studio 中選擇編輯以編輯您的狀態機資源。

   以下動畫展示了如何切換到工作流程 Studio 以編輯狀態機定義。
   說明如何在中使用工作流程工作室的動畫Application Composer。

   與工作流程 Studio 的整合以編輯中建立的狀態機器資源，Application Composer
   僅適用於`AWS::Serverless::StateMachine`資源。此整合不適用於使用
   資`AWS::StepFunctions::StateMachine`源的範本。

使用工作流程 Studio 中的CloudFormation定義替代動態參考資源

在工作流程 Studio 中，您可以在工作流程定CloudFormation義中使用定義替換來動態參考您已在 IaC 範本中定義的資源。您可以使用`${dollar_sign_brace}`符號將預留位置替代新增至工作流程定義，並在CloudFormation堆疊建立程序期間將它們取代為實際值。如需定義取代的詳細資訊，請參閱DefinitionSubstitutions 在AWS SAM範本中。

以下動畫展示了如何在狀態機定義中為資源添加佔位符替換。
一種動畫，說明當您在Application Composer中使用 Workflow Studio 時，如何動態參考資源 (例如 AWS Lambda函數、定義替代)。

將服務整合工作 Connect 至增強型元件卡

您可以將調用優化服務集成的任務連接到Application Composer畫布中的增強型組件卡。這樣做會自動對應工作流程定義中`${dollar_sign_brace}`符號所指定的任何預留位置替代，以

及StateMachine資源的DefinitionSubstitution屬性。它也會為狀態機新增適當的AWS SAM原則。

如果您將最佳化的服務整合工作與標準元件卡對應，則Application Composer畫布上不會顯示連線線。

以下動畫展示了如何將最佳化的工作連接至增強的元件卡，以及如何在「變更檢查器」中 Inspector 視變更。
一種動畫，說明如何在中使用 Workflow Studio 時，將呼叫最佳化服務整合的工作連接至增強型元件卡的工作Application Composer。

您無法使用增強型元件卡或標準元件卡的最佳化服務整合來連接工作狀態中的 AWSSDK 整合。對於這些工作，您可以在Application Composer畫布中對應「資源屬性」面板中的替代項目，並在AWS SAM範本中新增政策。

> **Tip**
>
> 或者，您也可以在「資源屬性」面板中的「定義替代」下，對應狀態機的預留位置替代。執行此操作時，您必須在狀態機器執行角色中AWS 服務為工作狀態呼叫新增必要的權限。如需執行角色可能需要之權限的相關資訊，請參閱工作流程中的執行角色。

下列動畫展示如何在「資源屬性」面板中手動更新預留位置取代對應。
一種動畫，說明當您在中使用 Workflow Studio 時，如何在「資源」屬性面板中手動更新預留位置替代對映Application Composer。

導入現有項目並在本地同步

您可以在Application Composer中開啟現有AWS SAM專案CloudFormation和專案，以便更好地瞭解和修改其設計。使用Application Composer的本地同步功能，您可以自動同步模板和代碼文件並將其保存到本地構建計算機。使用本地同步模式可以補充您現有的開發流程。請確定您的瀏覽器支援檔案系統存取 API，可讓 Web 應用程式在本機檔案系統中讀取、寫入及儲存檔案。我們建議使用 Google Chrome 或 Microsoft Edge。

不可用的工作流程 Studio 功能 AWS 應用程式編寫器

當您在中使用工作流程 Studio 時Application Composer，某些工作流程 Studio 功能無法使用。此外，面板中提供的「API 參數Inspector」區段支援CloudFormation定義替代。您可以程式碼模式使用${dollar_sign_brace}符號在中加入替代。如需此符號的詳細資訊，請參閱DefinitionSubstitutions 在AWS SAM範本中。

下列清單說明在中使用工作流程 Studio 時無法使用的工作流程 Studio 功能Application Composer：

- 入門範本 — 初學者範 ready-to-run 本是自動建立工作流程 proptotypes 和定義的範例專案。這些範本會將專案所需的所有相關AWS資源部署到AWS 帳戶.
- 設 Config 模式 — 此模式可讓您管理狀態機器的組態。您可以在 IaC 範本中更新您的狀態機器設定，或使用Application Composer畫布中的資源屬性面板。如需有關在「資源屬性」面板中更新組態的資訊，請參閱將服務整合工作 Connect 至增強型元件卡。
- TestState API
- 從 [工作流程 Studio] 的 [動作] 下拉式按鈕匯入或匯出工作流程定義的選項。而是從Application Composer功能表中選取「開啟」>「專案資料夾」。確保您已啟用本地同步模式，以將Application Composer畫布中的更改直接保存到本地計算機。
- 「執行」按鈕。當您使用工作流工作室中Application Composer，Application Composer生成您的工作流程的 IaC 代碼。因此，您必須先部署範本。然後，在主控台或透過執行工作流程AWS Command Line Interface(AWS CLI)。

使用建立 Step Functions 的 Lambda 狀態機 AWS CloudFormation

本教程將向您展示如何使用創建基本 AWS Lambda 函數 AWS CloudFormation。您將使用主 AWS CloudFormation 控台和 YAML 範本建立堆疊 (IAM 角色、Lambda 函數和狀態機器)。然後，您將使用 AWS Step Functions 控制台來啟動狀態機執行。

若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudFormation 使用 CloudFormation 範本和*AWS::StepFunctions::StateMachine*資源〉。

主題

- 步驟 1：設定 AWS CloudFormation 範本
- 步驟 2：使用 AWS CloudFormation 範本建立 Lambda 狀態機
- 步驟 3：啟動狀態機執行

步驟 1：設定 AWS CloudFormation 範本

在使用範例範本之前，建議您了解如何宣告 AWS CloudFormation 範本的不同部分。

主題

- 建立適用於 Lambda 的 IAM 角色

- 建立 Lambda 函式
- 若要建立狀態機器執行的 IAM 角色
- 若要建立 Lambda 狀態機

建立適用於 Lambda 的 IAM 角色

定義與 Lambda 函數的 IAM 角色相關聯的信任政策。下列範例會定義使用 YAML 或 JSON 的信任原則。

YAML

```
LambdaExecutionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Principal:
            Service: lambda.amazonaws.com
          Action: "sts:AssumeRole"
```

JSON

```
        "LambdaExecutionRole": {
          "Type": "AWS::IAM::Role",
          "Properties": {
              "AssumeRolePolicyDocument": {
                  "Version": "2012-10-17",
                  "Statement": [
                      {
                          "Effect": "Allow",
                          "Principal": {
                              "Service": "lambda.amazonaws.com"
                          },
                          "Action": "sts:AssumeRole"
                      }
                  ]
              }
          }
```

建立 Lambda 函式

為將列印訊息的 Lambda 函數定義下列屬性`Hello World`。

> **Important**
>
> 確保您的 Lambda 函數與狀態機位於相同的 AWS 帳戶和 AWS 區域內。

YAML

```
MyLambdaFunction:
  Type: "AWS::Lambda::Function"
  Properties:
    Handler: "index.handler"
    Role: !GetAtt [ LambdaExecutionRole, Arn ]
    Code:
      ZipFile: |
        exports.handler = (event, context, callback) => {
            callback(null, "Hello World!");
        };
    Runtime: "nodejs12.x"
    Timeout: "25"
```

JSON

```
        "MyLambdaFunction": {
            "Type": "AWS::Lambda::Function",
            "Properties": {
                "Handler": "index.handler",
                "Role": {
                    "Fn::GetAtt": [
                        "LambdaExecutionRole",
                        "Arn"
                    ]
                },
                "Code": {
                    "ZipFile": "exports.handler = (event, context, callback) => {\n
   callback(null, \"Hello World!\");\n};\n"
                },
                "Runtime": "nodejs12.x",
                "Timeout": "25"
```

```
            }
        },
```

若要建立狀態機器執行的 IAM 角色

定義與狀態機器執行的 IAM 角色相關聯的信任政策。

YAML

```
StatesExecutionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - !Sub states.${AWS::Region}.amazonaws.com
          Action: "sts:AssumeRole"
    Path: "/"
    Policies:
      - PolicyName: StatesExecutionPolicy
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - "lambda:InvokeFunction"
              Resource: "*"
```

JSON

```
        "StatesExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
```

```
                                "Service": [
                                    {
                                        "Fn::Sub": "states.
${AWS::Region}.amazonaws.com"
                                    }
                                ]
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "StatesExecutionPolicy",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "lambda:InvokeFunction"
                                    ],
                                    "Resource": "*"
                                }
                            ]
                        }
                    }
                ]
            }
        },
```

若要建立 Lambda 狀態機

定義 Lambda 狀態機。

YAML

```
MyStateMachine:
  Type: "AWS::StepFunctions::StateMachine"
  Properties:
    DefinitionString:
```

```
      !Sub
        - |-
          {
            "Comment": "A Hello World example using an AWS Lambda function",
            "StartAt": "HelloWorld",
            "States": {
              "HelloWorld": {
                "Type": "Task",
                "Resource": "${lambdaArn}",
                "End": true
              }
            }
          }
        - {lambdaArn: !GetAtt [ MyLambdaFunction, Arn ]}
    RoleArn: !GetAtt [ StatesExecutionRole, Arn ]
```

JSON

```
        "MyStateMachine": {
            "Type": "AWS::StepFunctions::StateMachine",
            "Properties": {
                "DefinitionString": {
                    "Fn::Sub": [
                        "{\n  \"Comment\": \"A Hello World example using an
 AWS Lambda function\",\n  \"StartAt\": \"HelloWorld\",\n  \"States\": {\n
 \"HelloWorld\": {\n      \"Type\": \"Task\",\n      \"Resource\": \"${lambdaArn}\",
\n      \"End\": true\n    }\n  }\n}",
                        {
                            "lambdaArn": {
                                "Fn::GetAtt": [
                                    "MyLambdaFunction",
                                    "Arn"
                                ]
                            }
                        }
                    ]
                },
                "RoleArn": {
                    "Fn::GetAtt": [
                        "StatesExecutionRole",
                        "Arn"
                    ]
                }
```

```
            }
        }
```

步驟 2：使用 AWS CloudFormation 範本建立 Lambda 狀態機

瞭解 AWS CloudFormation 範本的元件後，您就可以將它們放在一起，並使用範本建立 AWS CloudFormation 堆疊。

若要建立 Lambda 狀態機

1. 將下列範例資料複製到名為 `MyStateMachine.yaml` 的檔案中以用於 YAML 範例，或 `MyStateMachine.json` 以用於 JSON。

   YAML

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "An example template with an IAM role for a Lambda state machine."
Resources:
  LambdaExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: "sts:AssumeRole"

  MyLambdaFunction:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role: !GetAtt [ LambdaExecutionRole, Arn ]
      Code:
        ZipFile: |
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      Runtime: "nodejs12.x"
      Timeout: "25"
```

```
  StatesExecutionRole:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - !Sub states.${AWS::Region}.amazonaws.com
            Action: "sts:AssumeRole"
      Path: "/"
      Policies:
        - PolicyName: StatesExecutionPolicy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - "lambda:InvokeFunction"
                Resource: "*"

  MyStateMachine:
    Type: "AWS::StepFunctions::StateMachine"
    Properties:
      DefinitionString:
        !Sub
          - |-
            {
              "Comment": "A Hello World example using an AWS Lambda function",
              "StartAt": "HelloWorld",
              "States": {
                "HelloWorld": {
                  "Type": "Task",
                  "Resource": "${lambdaArn}",
                  "End": true
                }
              }
            }
          - {lambdaArn: !GetAtt [ MyLambdaFunction, Arn ]}
      RoleArn: !GetAtt [ StatesExecutionRole, Arn ]
```

JSON

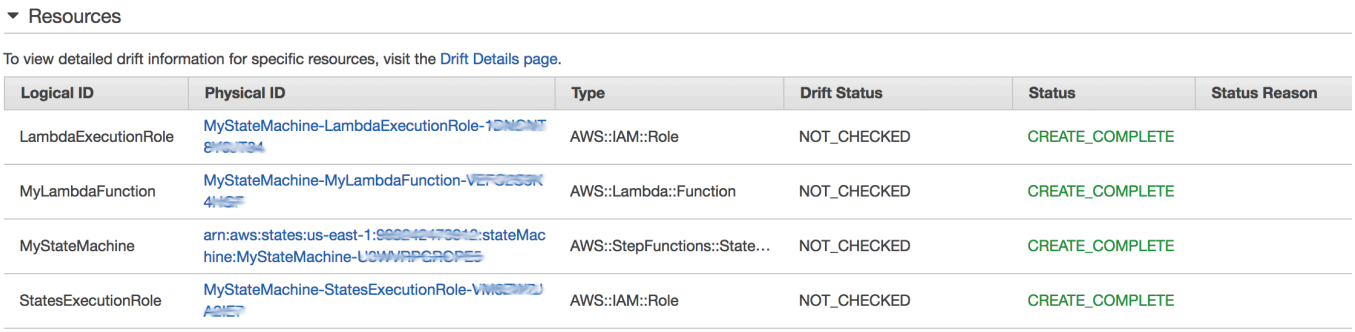
```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "An example template with an IAM role for a Lambda state
 machine.",
    "Resources": {
        "LambdaExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "lambda.amazonaws.com"
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                }
            }
        },
        "MyLambdaFunction": {
            "Type": "AWS::Lambda::Function",
            "Properties": {
                "Handler": "index.handler",
                "Role": {
                    "Fn::GetAtt": [
                        "LambdaExecutionRole",
                        "Arn"
                    ]
                },
                "Code": {
                    "ZipFile": "exports.handler = (event, context, callback) =>
 {\n    callback(null, \"Hello World!\");\n};\n"
                },
                "Runtime": "nodejs12.x",
                "Timeout": "25"
            }
        },
        "StatesExecutionRole": {
```

```
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    {
                                        "Fn::Sub": "states.
${AWS::Region}.amazonaws.com"
                                    }
                                ]
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "Path": "/",
                "Policies": [
                    {
                        "PolicyName": "StatesExecutionPolicy",
                        "PolicyDocument": {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Action": [
                                        "lambda:InvokeFunction"
                                    ],
                                    "Resource": "*"
                                }
                            ]
                        }
                    }
                ]
            }
        },
        "MyStateMachine": {
            "Type": "AWS::StepFunctions::StateMachine",
            "Properties": {
                "DefinitionString": {
                    "Fn::Sub": [
```

```
                        "{\n  \"Comment\": \"A Hello World example using
 an AWS Lambda function\",\n  \"StartAt\": \"HelloWorld\",\n  \"States\":
 {\n    \"HelloWorld\": {\n      \"Type\": \"Task\",\n      \"Resource\":
 \"${lambdaArn}\",\n      \"End\": true\n    }\n  }\n}",
                        {
                            "lambdaArn": {
                                "Fn::GetAtt": [
                                    "MyLambdaFunction",
                                    "Arn"
                                ]
                            }
                        }
                    ]
                },
                "RoleArn": {
                    "Fn::GetAtt": [
                        "StatesExecutionRole",
                        "Arn"
                    ]
                }
            }
        }
    }
}
```

2. 開啟 AWS CloudFormation 主控台，然後選擇 Create Stack (建立堆疊)。
3. 在 Select Template (選取範本) 頁面上，選擇 Upload a template to Amazon S3 (將範本上傳到 Amazon S3)。選擇您的 `MyStateMachine` 檔案，然後選擇 Next (下一步)。
4. 在 Specify Details (指定詳細資訊) 頁面上，為 Stack Name (堆疊名稱) 輸入 `MyStateMachine`，然後選擇 Next (下一步)。
5. 在選項頁面上，選擇下一步。
6. 在 [檢閱] 頁面上，選擇 [我確認 AWS CloudFormation 可能會建立 IAM 資源]。 ，然後選擇 [建立]。

   AWS CloudFormation 會開始建立`MyStateMachine`堆疊，並顯示「建立中 _ 進度」狀態。程序完成後， AWS CloudFormation 會顯示 CREATE_COMPLETE 狀態。
7. (選用) 若要顯示堆疊中的資源，請選取堆疊，然後選擇 Resources (資源) 標籤。



▼ Resources

To view detailed drift information for specific resources, visit the Drift Details page.

Logical ID	Physical ID	Type	Drift Status	Status	Status Reason
LambdaExecutionRole	MyStateMachine-LambdaExecutionRole-1[redacted]	AWS::IAM::Role	NOT_CHECKED	CREATE_COMPLETE	
MyLambdaFunction	MyStateMachine-MyLambdaFunction-[redacted]	AWS::Lambda::Function	NOT_CHECKED	CREATE_COMPLETE	
MyStateMachine	arn:aws:states:us-east-1:[redacted]:stateMachine:MyStateMachine-[redacted]	AWS::StepFunctions::State…	NOT_CHECKED	CREATE_COMPLETE	
StatesExecutionRole	MyStateMachine-StatesExecutionRole-[redacted]	AWS::IAM::Role	NOT_CHECKED	CREATE_COMPLETE	

步驟 3：啟動狀態機執行

建立 Lambda 狀態機器之後，您就可以開始執行該機器。

開始狀態機器執行

1. 開啟 Step Functions 主控台，然後選擇您使用建立的狀態機器名稱 AWS CloudFormation。
2. 在 ***MyStateMachine-ABCDEFGIJ*** 1K 頁面上，選擇新的執行。

   系統會隨即顯示 New execution (新執行) 頁面。
3. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。
4. 選擇 Start Execution (開始執行)。

   狀態機器會開始新執行，且隨即會出現新頁面顯示您正在執行的執行。
5. (選用) 在 Execution Details (執行詳細資訊) 中，檢閱 Execution Status (執行狀態)、Started (已開始) 和 Closed (已結束) 時間戳記。
6. 若要檢視執行結果，請選擇 Output (輸出)。

建立要Step Functions使用的Lambda狀態機 AWS CDK

本教學課程說明如何建AWS Step Functions立包含使用的AWS Lambda函數的狀態機器AWS Cloud Development Kit (AWS CDK)。這AWS CDK是一個基礎設施即代碼（IAC）框架，可以讓你使用一個完整的編程語言定義 AWS 基礎設施。您可以使用其中一種支援的語言撰寫應用程式，其中包含一或多個堆疊。CDK然後，您可以將其合成為AWS CloudFormation模板並將其部署到您的 AWS 帳戶。我們將使用此方法來定義包含Lambda函數的Step Functions狀態機器，然後使用 AWS Management Console 來運行狀態機器。

在開始本教學課程之前，您必須依照開AWS CDK發AWS Cloud Development Kit (AWS CDK) 人員指南中的 [AWS CDK-先決條件] 中所述設定您的開發環境。然後，AWS CDK使用以下命令安裝在AWS CLI：

```
npm install -g aws-cdk
```

本自學課程產生的結果與the section called “使用建立 Lambda 狀態機器 AWS CloudFormation”. 但是，在本教程中，AWS CDK不需要您創建任何IAM角色; 為您AWS CDK做到了。該AWS CDK版本還包括一個Succeed步驟來說明如何向狀態機添加其他步驟。

> **Tip**
>
> 若要將啟動工作流程使用的範例無伺服器應AWS CDK用Step Functions程式部署 TypeScript 至您的工作流程 AWS 帳戶，請參閱 AWS Step Functions 工作坊AWS CDK的單元 10-部署。

主題

步驟 1：設定您的 AWS CDK 專案

1. 如果您願意，請在主目錄或其他目錄中執行下列命令，為新AWS CDK應用程式建立目錄。

> **Important**
>
> 請務必命名目錄`step`。AWS CDK應用程式範本會使用目錄名稱來產生來源檔案和類別的名稱。若使用不同名稱，您的應用程式將與本教學課程不相符。

TypeScript

```
mkdir step && cd step
```

JavaScript

```
mkdir step && cd step
```

Python

```
mkdir step && cd step
```

Java

```
mkdir step && cd step
```

C#

確保您已安裝 .NET 6.0 或更高版本。如需資訊，請參閱支援的版本。

```
mkdir step && cd step
```

2. 使用 cdk init 指令初始化應用程式。指定所需的範本（「app」）和程式設計語言，如下列範例所示。

TypeScript

```
cdk init --language typescript
```

JavaScript

```
cdk init --language javascript
```

Python

```
cdk init --language python
```

專案初始化後，啟動專案的虛擬環境並安裝AWS CDK的基準相依性。

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
cdk init --language java
```

C#

```
cdk init --language csharp
```

步驟 2：用AWS CDK來建立狀態機

首先，我們將介紹定義Lambda函數和Step Functions狀態機器的各個代碼片段。然後，我們將解釋如何將它們放在您的AWS CDK應用程序中。最後，您將看到如何合成和部署這些資源。

建立 Lambda 函數

下列AWS CDK程式碼會定義Lambda函式，並以內嵌方式提供其原始程式碼。

TypeScript

```
const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
    code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
```

```
    runtime: lambda.Runtime.NODEJS_18_X,
    handler: "index.handler",
    timeout: cdk.Duration.seconds(3)
});
```

JavaScript

```
const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
    code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
    runtime: lambda.Runtime.NODEJS_18_X,
    handler: "index.handler",
    timeout: cdk.Duration.seconds(3)
});
```

Python

```
hello_function = lambda_.Function(
            self, "MyLambdaFunction",
            code=lambda_.Code.from_inline("""
            exports.handler = (event, context, callback) => {
                callback(null, "Hello World!");
                }"""),
                runtime=lambda_.Runtime.NODEJS_18_X,
                handler="index.handler",
                timeout=Duration.seconds(25))
```

Java

```
final Function helloFunction = Function.Builder.create(this, "MyLambdaFunction")
        .code(Code.fromInline(
                "exports.handler = (event, context, callback) => { callback(null,
 'Hello World!' );}"))
        .runtime(Runtime.NODEJS_18_X)
        .handler("index.handler")
        .timeout(Duration.seconds(25))
        .build();
```

C#

```
var helloFunction = new Function(this, "MyLambdaFunction", new FunctionProps
{
    Code = Code.FromInline(@"`
      exports.handler = (event, context, callback) => {
        callback(null, 'Hello World!');
      }"),
    Runtime = Runtime.NODEJS_12_X,
    Handler = "index.handler",
    Timeout = Duration.Seconds(25)
});
```

你可以在這個簡短的示例代碼中看到：

- 函數的邏輯名稱，MyLambdaFunction.
- 函數的原始程式碼，以字串形式嵌入到應用程式的原始AWS CDK程式碼中。
- 其他函數屬性，例如要使用的執行階段 (節點 18.x)、函數的進入點和逾時。

建立 狀態機器

我們的狀態機有兩種狀態：Lambda函數任務和Succeed狀態。該函數要求我們創建一個調Step Functionsthe section called “任務”用我們的函數。此任務狀態用作狀態的狀態機器中的第一個步驟。成功狀態會使用工作狀態的next()方法新增至狀態機器。下列程式碼會先叫用名為的函數MyLambdaTask，然後使用該next()方法定義名為GreetedWorld的成功狀態。

TypeScript

```
const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
  definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
    lambdaFunction: helloFunction
  }).next(new sfn.Succeed(this, "GreetedWorld"))
});
```

JavaScript

```
const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
  definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
    lambdaFunction: helloFunction
```

```
  }).next(new sfn.Succeed(this, "GreetedWorld"))
});
```

Python

```
state_machine = sfn.StateMachine(
                                  self, "MyStateMachine",
                                  definition=tasks.LambdaInvoke(
                                  self, "MyLambdaTask",
                                  lambda_function=hello_function)
                                  .next(sfn.Succeed(self, "GreetedWorld")))
```

Java

```
final StateMachine stateMachine = StateMachine.Builder.create(this,
 "MyStateMachine")
        .definition(LambdaInvoke.Builder.create(this, "MyLambdaTask")
            .lambdaFunction(helloFunction)
            .build()
            .next(new Succeed(this, "GreetedWorld")))
        .build();
```

C#

```
var stateMachine = new StateMachine(this, "MyStateMachine", new StateMachineProps {
    DefinitionBody = DefinitionBody.FromChainable(new LambdaInvoke(this,
 "MyLambdaTask", new LambdaInvokeProps
    {
        LambdaFunction = helloFunction
    })
    .Next(new Succeed(this, "GreetedWorld")))
});
```

若要建置和部署AWS CDK應用程式

在新建立的AWS CDK專案中，編輯包含堆疊定義的檔案，使其看起來像下列範例程式碼。您將從前面的章節中識別Lambda函數和Step Functions狀態機的定義。

1. 更新堆疊，如下列範例所示。

TypeScript

lib/step-stack.ts使用下面的代碼更新。

```
import * as cdk from 'aws-cdk-lib';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as sfn from 'aws-cdk-lib/aws-stepfunctions';
import * as tasks from 'aws-cdk-lib/aws-stepfunctions-tasks';

export class StepStack extends cdk.Stack {
  constructor(app: cdk.App, id: string) {
    super(app, id);

    const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
      code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
      runtime: lambda.Runtime.NODEJS_18_X,
      handler: "index.handler",
      timeout: cdk.Duration.seconds(3)
    });

    const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
      definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
        lambdaFunction: helloFunction
      }).next(new sfn.Succeed(this, "GreetedWorld"))
    });
  }
}
```

JavaScript

lib/step-stack.js使用下面的代碼更新。

```
import * as cdk from 'aws-cdk-lib';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as sfn from 'aws-cdk-lib/aws-stepfunctions';
import * as tasks from 'aws-cdk-lib/aws-stepfunctions-tasks';

export class StepStack extends cdk.Stack {
```

```
  constructor(app, id) {
    super(app, id);

    const helloFunction = new lambda.Function(this, 'MyLambdaFunction', {
      code: lambda.Code.fromInline(`
          exports.handler = (event, context, callback) => {
              callback(null, "Hello World!");
          };
      `),
      runtime: lambda.Runtime.NODEJS_18_X,
      handler: "index.handler",
      timeout: cdk.Duration.seconds(3)
    });

    const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
      definition: new tasks.LambdaInvoke(this, "MyLambdaTask", {
        lambdaFunction: helloFunction
      }).next(new sfn.Succeed(this, "GreetedWorld"))
    });
  }
}
```

Python

step/step_stack.py使用下面的代碼更新。

```
from aws_cdk import (
    Duration,
    Stack,
    aws_stepfunctions as sfn,
    aws_stepfunctions_tasks as tasks,
    aws_lambda as lambda_
)
class StepStack(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        hello_function = lambda_.Function(
            self, "MyLambdaFunction",
            code=lambda_.Code.from_inline("""
            exports.handler = (event, context, callback) => {
                callback(null, "Hello World!");
```

```
                }"""),
                runtime=lambda_.Runtime.NODEJS_18_X,
                handler="index.handler",
                timeout=Duration.seconds(25))

        state_machine = sfn.StateMachine(
            self, "MyStateMachine",
            definition=tasks.LambdaInvoke(
            self, "MyLambdaTask",
            lambda_function=hello_function)
            .next(sfn.Succeed(self, "GreetedWorld")))
```

Java

src/main/java/com.myorg/StepStack.java使用下面的代碼更新。

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;
import software.amazon.awscdk.Duration;
import software.amazon.awscdk.services.lambda.Code;
import software.amazon.awscdk.services.lambda.Function;
import software.amazon.awscdk.services.lambda.Runtime;
import software.amazon.awscdk.services.stepfunctions.StateMachine;
import software.amazon.awscdk.services.stepfunctions.Succeed;
import software.amazon.awscdk.services.stepfunctions.tasks.LambdaInvoke;

public class StepStack extends Stack {
    public StepStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public StepStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        final Function helloFunction = Function.Builder.create(this,
 "MyLambdaFunction")
                .code(Code.fromInline(
                        "exports.handler = (event, context, callback) =>
 { callback(null, 'Hello World!' );}"))
```

```
                .runtime(Runtime.NODEJS_18_X)
                .handler("index.handler")
                .timeout(Duration.seconds(25))
                .build();

        final StateMachine stateMachine = StateMachine.Builder.create(this,
 "MyStateMachine")
                .definition(LambdaInvoke.Builder.create(this, "MyLambdaTask")
                        .lambdaFunction(helloFunction)
                        .build()
                        .next(new Succeed(this, "GreetedWorld")))
                .build();
    }
}
```

C#

scr/Step/StepStack.cs使用下面的代碼更新。

```
using Amazon.CDK;
using Constructs;
using Amazon.CDK.AWS.Lambda;
using Amazon.CDK.AWS.StepFunctions;
using Amazon.CDK.AWS.StepFunctions.Tasks;

namespace Step
{
    public class StepStack : Stack
    {
        internal StepStack(Construct scope, string id, IStackProps props =
 null) : base(scope, id, props)
        {
            var helloFunction = new Function(this, "MyLambdaFunction", new
 FunctionProps
            {
                Code = Code.FromInline(@"exports.handler = (event, context,
 callback) => {
                    callback(null, 'Hello World!');
                }"),
                Runtime = Runtime.NODEJS_18_X,
                Handler = "index.handler",
                Timeout = Duration.Seconds(25)
            });
```

```
            var stateMachine = new StateMachine(this, "MyStateMachine", new
 StateMachineProps
            {
                DefinitionBody = DefinitionBody.FromChainable(new
 LambdaInvoke(this, "MyLambdaTask", new LambdaInvokeProps
                {
                    LambdaFunction = helloFunction
                })
                .Next(new Succeed(this, "GreetedWorld")))
            });
        }
    }
}
```

2. 保存源文件，然後在應用程序的主目錄中運行`cdk synth`命令。

   AWS CDK運行該應用程序並從中合成AWS CloudFormation模板。 AWS CDK然後顯示範本。

   > **Note**
   >
   > 如果您曾經 TypeScript 建立AWS CDK專案，執行命`cdk synth`令可能會傳回下列錯誤。
   >
   > ```
   > TSError: # Unable to compile TypeScript:
   > bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
   > ```
   >
   > 修改`bin/step.ts`檔案，如下列範例所示，以解決此錯誤。
   >
   > ```
   > #!/usr/bin/env node
   > import 'source-map-support/register';
   > import * as cdk from 'aws-cdk-lib';
   > import { StepStack } from '../lib/step-stack';
   >
   > const app = new cdk.App();
   > new StepStack(app, 'StepStack');
   > app.synth();
   > ```

3. 若要將 Lambda 函數和 Step Functions 數狀態機器部署到您的 AWS 帳戶，請發出問題`cdk deploy`。系統會要求您核准 AWS CDK 已產生的 IAM 政策。

步驟 3：啟動狀態機執行

建立狀態機之後，您就可以開始執行它。

開始狀態機器執行

1. 開啟 Step Functions 主控台，然後選擇您使用建立的狀態機器名稱AWS CDK。
2. 在 [狀態機器] 頁面上，選擇 [開始執行]。

   此時會顯示「開始執行」對話方塊。
3. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。
4. 選擇 Start Execution (開始執行)。

   您的狀態機的執行開始，並顯示一個顯示正在運行執行的新頁面。
5. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

步驟 4：清除

測試完狀態機之後，我們建議您移除狀態機和相關 Lambda 函數，以釋放 AWS 帳戶. 在應用程序的主目錄中運行該`cdk destroy`命令以刪除狀態機。

後續步驟

若要深入瞭解如何使用開發 AWS 基礎架構AWS CDK，請參閱AWS CDK開發人員指南。

如需有關以所選語言編寫 AWS CDK 應用程式的資訊，請參閱：

TypeScript

AWS CDK在中使用 TypeScript

JavaScript

AWS CDK在中使用 JavaScript

Python

AWS CDK在 Python 中使用

Java

AWS CDK在 Java 中使用

C#

AWS CDK在 C＃ 中使用

如需本教學課程中使用之「 AWS 建構程式庫」模組的詳細資訊，請參閱下列 AWS CDK API 參考概觀：

- AWS-拉姆達
- AWS-步驟函數
- aws-stepfunctions-tasks

使用同步快速狀態機器建立 API Gateway REST API AWS CDK

本教學課程說明如何建立具有同步快速狀態機的 API Gateway REST API，做為使用 AWS Cloud Development Kit (AWS CDK). 本教程將使用該`StepFunctionsRestApi`構造將狀態機連接到 API Gateway。該`StepFunctionsRestApi`構造將設置一個默認的輸入/輸出映射和 API Gateway REST API，具有所需的權限和 HTTP「任何」方法。這 AWS CDK 是一個基礎設施即代碼（IAC）框架，可以讓你使用一個完整的編程語言定義 AWS 基礎設施。您可以使用 CDK 支持的語言之一編寫應用程序，包含一個或多個堆棧，然後將其合成到 AWS CloudFormation 模板並將其部署到您 AWS 的帳戶中。我們將使用它來定義 API Gateway REST API，該 API 與同步快速狀態機整合為後端，然後使用 AWS Management Console 來啟動執行。

在開始學習本教學課程之前，請 AWS CDK 依照入門使用 AWS CDK -先決條件中所述設定您的開發環境，然後發出以下指令來安裝：

```
npm install -g aws-cdk
```

主題

步驟 1：設定您的 AWS CDK 專案

首先，為您的新 AWS CDK 應用程序創建一個目錄並初始化項目。

TypeScript

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language typescript
```

JavaScript

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language javascript
```

Python

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language python
```

專案初始化後，啟動專案的虛擬環境並安裝 AWS CDK的基準相依性。

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir stepfunctions-rest-api
```

```
cd stepfunctions-rest-api
cdk init --language java
```

C#

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language csharp
```

Go

```
mkdir stepfunctions-rest-api
cd stepfunctions-rest-api
cdk init --language go
```

> **Note**
>
> 請務必命名目錄`stepfunctions-rest-api`。 AWS CDK 應用程式範本會使用目錄名稱來產生來源檔案和類別的名稱。若使用不同名稱，您的應用程式將與本教學課程不相符。

現在安裝構造庫模塊 AWS Step Functions 和 Amazon API Gateway。

TypeScript

```
npm install @aws-cdk/aws-stepfunctions @aws-cdk/aws-apigateway
```

JavaScript

```
npm install @aws-cdk/aws-stepfunctions @aws-cdk/aws-apigateway
```

Python

```
python -m pip install aws-cdk.aws-stepfunctions
python -m pip install aws-cdk.aws-apigateway
```

Java

編輯專案的 `pom.xml`，在現有的 `<dependencies>` 容器內新增下列相依性。

```
        <dependency>
            <groupId>software.amazon.awscdk</groupId>
            <artifactId>stepfunctions</artifactId>
            <version>${cdk.version}</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awscdk</groupId>
            <artifactId>apigateway</artifactId>
            <version>${cdk.version}</version>
        </dependency>
```

Maven 會在您下次建立應用程式時自動安裝這些相依性。若要建置，發行 `mvn compile` 或使用您的 Java IDE 的 Build 命令。

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

您也可以使用 Visual Studio NuGet GUI 安裝指定的套件，可透過 [工具] > [Package 件管理員] > NuGet [管理解決方案的 NuGet 套件] 取得。

安裝完模塊後，您可以通過導入以下軟件包在 AWS CDK 應用程序中使用它們。

TypeScript

```
@aws-cdk/aws-stepfunctions
@aws-cdk/aws-apigateway
```

JavaScript

```
@aws-cdk/aws-stepfunctions
@aws-cdk/aws-apigateway
```

Python

```
aws_cdk.aws_stepfunctions
aws_cdk.aws_apigateway
```

Java

```
software.amazon.awscdk.services.apigateway.StepFunctionsRestApi
software.amazon.awscdk.services.stepfunctions.Pass
software.amazon.awscdk.services.stepfunctions.StateMachine
software.amazon.awscdk.services.stepfunctions.StateMachineType
```

C#

```
Amazon.CDK.AWS.StepFunctions
Amazon.CDK.AWS.APIGateway
```

Go

將以下內容添加到`import`內部`stepfunctions-rest-api.go`。

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

步驟 2：使用建立具有同步快速狀態機後端整合的 API Gateway REST API AWS CDK

首先，我們將介紹定義同步快速狀態機器和 API Gateway REST API 的個別程式碼片段，然後說明如何將它們放在您的 AWS CDK 應用程式中。然後，您將看到如何合成和部署這些資源。

> **Note**
>
> 我們將在這裡展示的狀態機將是一個簡單的狀態機`Pass`狀態。

建立快速狀態機

這是定義具有狀態的簡單狀態機的 AWS CDK `Pass`代碼。

TypeScript

```
const machineDefinition = new stepfunctions.Pass(this, 'PassState', {
    result: {value:"Hello!"},
})
```

```
const stateMachine = new stepfunctions.StateMachine(this, 'MyStateMachine', {
    definition: machineDefinition,
    stateMachineType: stepfunctions.StateMachineType.EXPRESS,
});
```

JavaScript

```
const machineDefinition = new sfn.Pass(this, 'PassState', {
    result: {value:"Hello!"},
})

const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
    definition: machineDefinition,
    stateMachineType: stepfunctions.StateMachineType.EXPRESS,
});
```

Python

```
machine_definition = sfn.Pass(self,"PassState",
                        result = sfn.Result("Hello"))

state_machine = sfn.StateMachine(self, 'MyStateMachine',
        definition = machine_definition,
        state_machine_type = sfn.StateMachineType.EXPRESS)
```

Java

```
Pass machineDefinition = Pass.Builder.create(this, "PassState")
                        .result(Result.fromString("Hello"))
                        .build();

StateMachine stateMachine = StateMachine.Builder.create(this, "MyStateMachine")
                            .definition(machineDefinition)
                            .stateMachineType(StateMachineType.EXPRESS)
                            .build();
```

C#

```
var machineDefinition = new Pass(this, "PassState", new PassProps
{
    Result = Result.FromString("Hello")
```

```
});

var stateMachine = new StateMachine(this, "MyStateMachine", new StateMachineProps
{
    Definition = machineDefinition,
    StateMachineType = StateMachineType.EXPRESS
});
```

Go

```
var machineDefinition = awsstepfunctions.NewPass(stack, jsii.String("PassState"),
 &awsstepfunctions.PassProps
{
    Result: awsstepfunctions.NewResult(jsii.String("Hello")),
})

var stateMachine = awsstepfunctions.NewStateMachine(stack,
 jsii.String("StateMachine"), &awsstepfunctions.StateMachineProps
{
    Definition: machineDefinition,
    StateMachineType: awsstepfunctions.StateMachineType_EXPRESS,
})
```

您可以在此簡短的程式碼片段中看到：

- 名為的機器定義PassState，這是一個Pass狀態。
- 狀態機的邏輯名稱，MyStateMachine.
- 機器定義用作狀態機定義。
- 狀態機器類型設定為，EXPRESS因StepFunctionsRestApi為只允許同步快速狀態機。

若要使用建**StepFunctionsRestApi**構建立 API Gateway REST API

我們將使用StepFunctionsRestApi構造來創建具有所需權限和默認輸入/輸出映射的 API Gateway REST API。

TypeScript

```
const api = new apigateway.StepFunctionsRestApi(this,
  'StepFunctionsRestApi', { stateMachine: stateMachine });
```

JavaScript

```
const api = new apigateway.StepFunctionsRestApi(this,
  'StepFunctionsRestApi', { stateMachine: stateMachine });
```

Python

```
api = apigw.StepFunctionsRestApi(self, "StepFunctionsRestApi",
                                 state_machine = state_machine)
```

Java

```
StepFunctionsRestApi api = StepFunctionsRestApi.Builder.create(this,
 "StepFunctionsRestApi")
                            .stateMachine(stateMachine)
                            .build();
```

C#

```
var api = new StepFunctionsRestApi(this, "StepFunctionsRestApi", new
 StepFunctionsRestApiProps
{
    StateMachine = stateMachine
});
```

Go

```
awsapigateway.NewStepFunctionsRestApi(stack, jsii.String("StepFunctionsRestApi"),
 &awsapigateway.StepFunctionsRestApiProps
{
    StateMachine = stateMachine,
})
```

若要建置和部署 AWS CDK 應用程式

在您建立的 AWS CDK 專案中，編輯包含堆疊定義的檔案，使其看起來像下面的程式碼一樣。您將從上面識別 Step Functions 狀態機和 API Gateway 的定義。

TypeScript

更新 `lib/stepfunctions-rest-api-stack.ts`，讀取如下。

```
import * as cdk from 'aws-cdk-lib';
import * as stepfunctions from 'aws-cdk-lib/aws-stepfunctions'
import * as apigateway from 'aws-cdk-lib/aws-apigateway';


export class StepfunctionsRestApiStack extends cdk.Stack {
    constructor(scope: cdk.App, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const machineDefinition = new stepfunctions.Pass(this, 'PassState', {
        result: {value:"Hello!"},
    });

    const stateMachine = new stepfunctions.StateMachine(this, 'MyStateMachine', {
        definition: machineDefinition,
        stateMachineType: stepfunctions.StateMachineType.EXPRESS,
    });

    const api = new apigateway.StepFunctionsRestApi(this,
        'StepFunctionsRestApi', { stateMachine: stateMachine });
```

JavaScript

更新 `lib/stepfunctions-rest-api-stack.js`，讀取如下。

```
const cdk = require('@aws-cdk/core');
const stepfunctions = require('@aws-cdk/aws-stepfunctions');
const apigateway = require('@aws-cdk/aws-apigateway');


class StepfunctionsRestApiStack extends cdk.Stack {
    constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const machineDefinition = new stepfunctions.Pass(this, "PassState", {
        result: {value:"Hello!"},
    })

    const stateMachine = new sfn.StateMachine(this, 'MyStateMachine', {
        definition: machineDefinition,
        stateMachineType: stepfunctions.StateMachineType.EXPRESS,
    });
```

```
    const api = new apigateway.StepFunctionsRestApi(this,
        'StepFunctionsRestApi', { stateMachine: stateMachine });

    }
}

module.exports = { StepStack }
```

Python

更新 stepfunctions_rest_api/stepfunctions_rest_api_stack.py，讀取如下。

```
from aws_cdk import App, Stack
from constructs import Construct
from aws_cdk import aws_stepfunctions as sfn
from aws_cdk import aws_apigateway as apigw

class StepfunctionsRestApiStack(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)


        machine_definition = sfn.Pass(self,"PassState",
                                result = sfn.Result("Hello"))

        state_machine = sfn.StateMachine(self, 'MyStateMachine',
                definition = machine_definition,
                state_machine_type = sfn.StateMachineType.EXPRESS)

        api = apigw.StepFunctionsRestApi(self,
                    "StepFunctionsRestApi",
                    state_machine = state_machine)
```

Java

更新 src/main/java/com.myorg/StepfunctionsRestApiStack.java，讀取如下。

```
package com.myorg;


import software.amazon.awscdk.core.Construct;
import software.amazon.awscdk.core.Stack;
```

```
import software.amazon.awscdk.core.StackProps;
import software.amazon.awscdk.services.stepfunctions.Pass;
import software.amazon.awscdk.services.stepfunctions.StateMachine;
import software.amazon.awscdk.services.stepfunctions.StateMachineType;
import software.amazon.awscdk.services.apigateway.StepFunctionsRestApi;

public class StepfunctionsRestApiStack extends Stack {
    public StepfunctionsRestApiStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public StepfunctionsRestApiStack(final Construct scope, final String id, final
 StackProps props) {
        super(scope, id, props);

        Pass machineDefinition = Pass.Builder.create(this, "PassState")
                                .result(Result.fromString("Hello"))
                                .build();

        StateMachine stateMachine = StateMachine.Builder.create(this,
 "MyStateMachine")
                                    .definition(machineDefinition)
                                    .stateMachineType(StateMachineType.EXPRESS)
                                    .build();

        StepFunctionsRestApi api = StepFunctionsRestApi.Builder.create(this,
 "StepFunctionsRestApi")
                                   .stateMachine(stateMachine)
                                   .build();

    }
}
```

C#

更新 `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs`，讀取如下。

```
using Amazon.CDK;
using Amazon.CDK.AWS.StepFunctions;
using Amazon.CDK.AWS.APIGateway;

namespace StepfunctionsRestApi
{
    public class StepfunctionsRestApiStack : Stack
```

```
    {
        internal StepfunctionsRestApi(Construct scope, string id, IStackProps props
 = null) : base(scope, id, props)
        {
            var machineDefinition = new Pass(this, "PassState", new PassProps
            {
                Result = Result.FromString("Hello")
            });

            var stateMachine = new StateMachine(this, "MyStateMachine", new
 StateMachineProps
            {
                Definition = machineDefinition,
                StateMachineType = StateMachineType.EXPRESS
            });

            var api = new StepFunctionsRestApi(this, "StepFunctionsRestApi", new
 StepFunctionsRestApiProps
            {
                StateMachine = stateMachine
            });

        }
    }
}
```

Go

更新 `stepfunctions-rest-api.go`，讀取如下。

```
package main
import (
    "github.com/aws/aws-cdk-go/awscdk"
    "github.com/aws/aws-cdk-go/awscdk/awsapigateway"
    "github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
    "github.com/aws/constructs-go/constructs/v3"
    "github.com/aws/jsii-runtime-go"
)


type StepfunctionsRestApiGoStackProps struct {
    awscdk.StackProps
}
```

```
func NewStepfunctionsRestApiGoStack(scope constructs.Construct, id string, props
 *StepfunctionsRestApiGoStackProps) awscdk.Stack {
    var sprops awscdk.StackProps
    if props != nil {
        sprops = props.StackProps
    }
    stack := awscdk.NewStack(scope, &id, &sprops)

    // The code that defines your stack goes here
    var machineDefinition = awsstepfunctions.NewPass(stack,
 jsii.String("PassState"), &awsstepfunctions.PassProps
    {
        Result: awsstepfunctions.NewResult(jsii.String("Hello")),
    })

    var stateMachine = awsstepfunctions.NewStateMachine(stack,
 jsii.String("StateMachine"), &awsstepfunctions.StateMachineProps{
        Definition: machineDefinition,
        StateMachineType: awsstepfunctions.StateMachineType_EXPRESS,
    });

    awsapigateway.NewStepFunctionsRestApi(stack,
 jsii.String("StepFunctionsRestApi"), &awsapigateway.StepFunctionsRestApiProps{
        StateMachine = stateMachine,
    })

    return stack
}

func main() {
    app := awscdk.NewApp(nil)

    NewStepfunctionsRestApiGoStack(app, "StepfunctionsRestApiGoStack",
 &StepfunctionsRestApiGoStackProps{
        awscdk.StackProps{
            Env: env(),
        },
    })

    app.Synth(nil)
}

// env determines the AWS environment (account+region) in which our stack is to
```

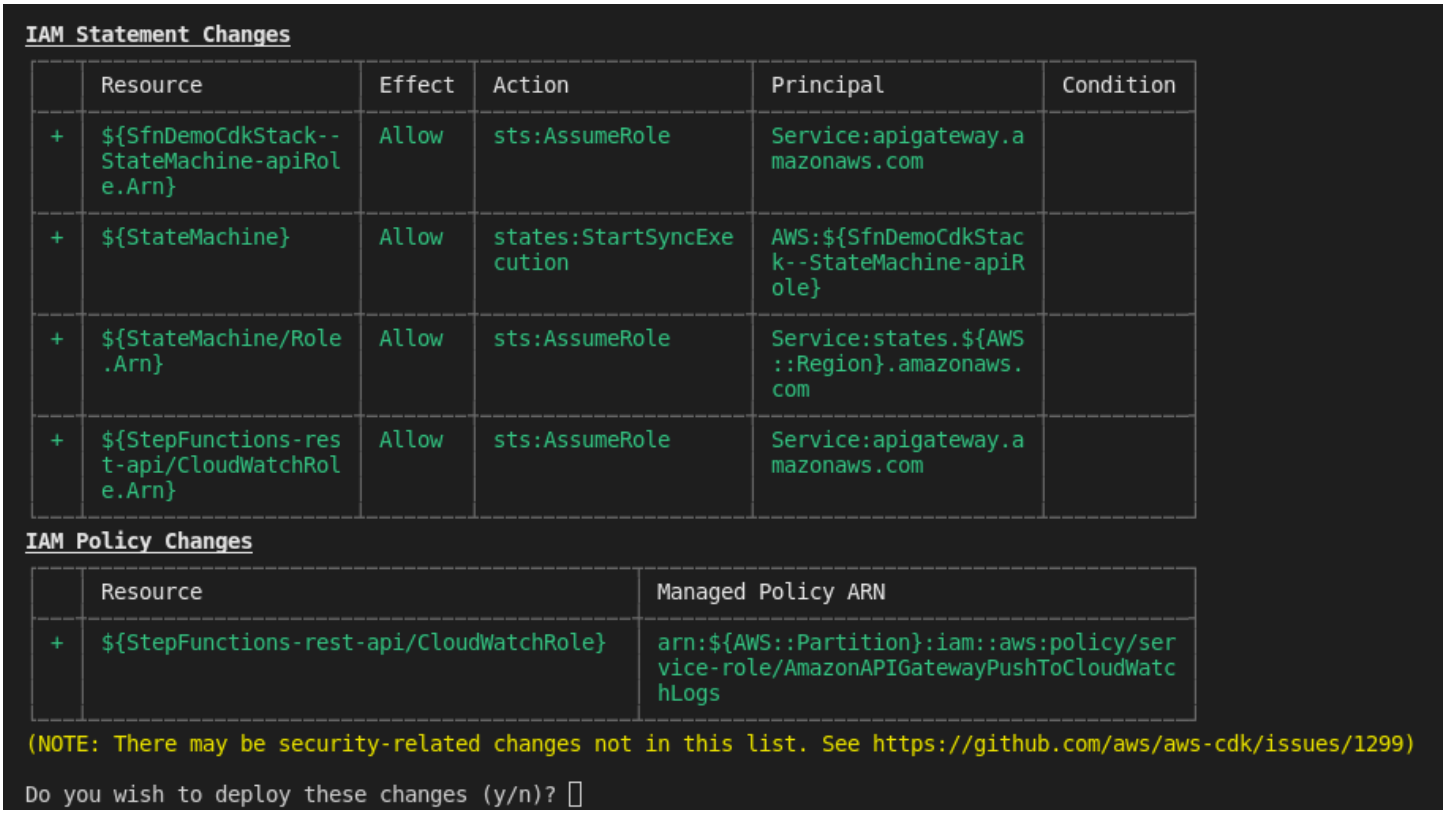
```
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
func env() *awscdk.Environment {
    // If unspecified, this stack will be "environment-agnostic".
    // Account/Region-dependent features and context lookups will not work, but a
    // single synthesized template can be deployed anywhere.
    //---------------------------------------------------------------------------
    return nil

    // Uncomment if you know exactly what account and region you want to deploy
    // the stack to. This is the recommendation for production stacks.
    //---------------------------------------------------------------------------
    // return &awscdk.Environment{
    //  Account: jsii.String("123456789012"),
    //  Region:  jsii.String("us-east-1"),
    // }

    // Uncomment to specialize this stack for the AWS Account and Region that are
    // implied by the current CLI configuration. This is recommended for dev
    // stacks.
    //---------------------------------------------------------------------------
    // return &awscdk.Environment{
    //  Account: jsii.String(os.Getenv("CDK_DEFAULT_ACCOUNT")),
    //  Region:  jsii.String(os.Getenv("CDK_DEFAULT_REGION")),
    // }
}
```

保存來源檔案，然後在應用程式的主目錄中發行 `cdk synth`。 AWS CDK 運行應用程序並從中合成 AWS CloudFormation 模板，然後顯示模板。

若要將 Amazon API Gateway 和 AWS Step Functions 狀態機器實際部署到您的 AWS 帳戶，請發出問題`cdk deploy`。系統會要求您核准 AWS CDK 已產生的 IAM 政策。正在建立的政策看起來會像這樣：



IAM Statement Changes

	Resource	Effect	Action	Principal	Condition
+	${SfnDemoCdkStack--StateMachine-apiRole.Arn}	Allow	sts:AssumeRole	Service:apigateway.amazonaws.com	
+	${StateMachine}	Allow	states:StartSyncExecution	AWS:${SfnDemoCdkStack--StateMachine-apiRole}	
+	${StateMachine/Role.Arn}	Allow	sts:AssumeRole	Service:states.${AWS::Region}.amazonaws.com	
+	${StepFunctions-rest-api/CloudWatchRole.Arn}	Allow	sts:AssumeRole	Service:apigateway.amazonaws.com	

IAM Policy Changes

	Resource	Managed Policy ARN
+	${StepFunctions-rest-api/CloudWatchRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs

(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)

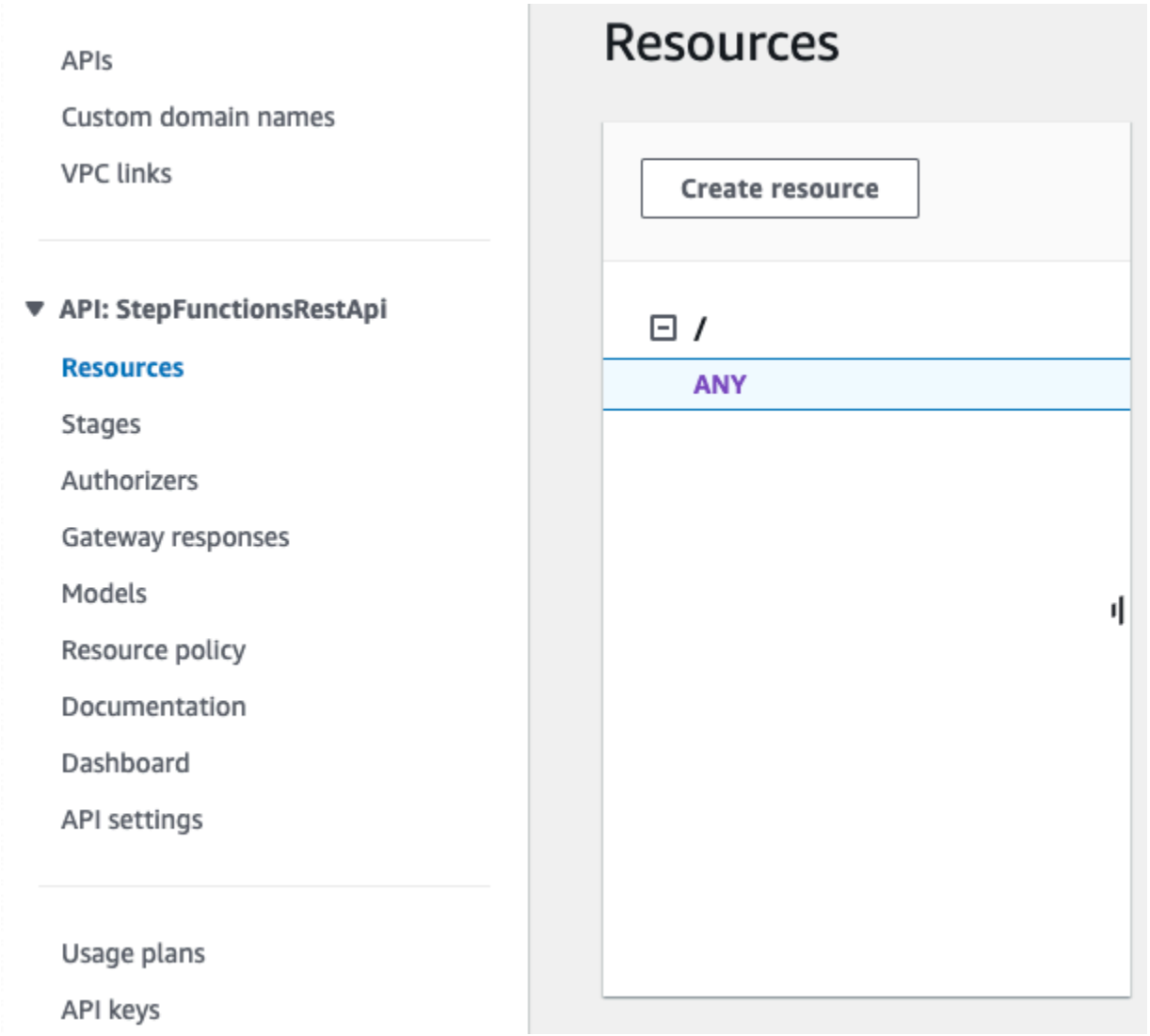
Do you wish to deploy these changes (y/n)?

步驟 3：測試 API Gateway

使用同步快速狀態機做為後端整合建立 API Gateway REST API 之後，您可以測試 API Gateway。

使用 API Gateway 主控台測試已部署的 API Gateway

1. 開啟 Amazon API Gateway 主控台並登入。
2. 選擇名為的其他 API `StepFunctionsRestApi`。
3. 在「資源」窗格中，選擇ANY方法。



4. 選擇測試標籤。您可能需要選擇向右箭頭按鈕才能顯示此索引標籤。
5. 針對 Method (方法) 選擇 POST。
6. 針對要求主體，複製下列要求參數。

```
{
    "key": "Hello"
}
```

7. 選擇 Test (測試)。下列資訊會隨即顯示：

   - Request (請求) 是針對方法所呼叫的資源路徑。
   - Status (狀態) 是回應的 HTTP 狀態碼。
   - Latency (延遲) 是從發起人收到請求到傳回回應之間的時間。

- 回應內文是 HTTP 回應內文。
- 回應標頭是 HTTP 回應標頭。
- CloudWatch 日誌顯示如果在 API Gateway 主控台外呼叫此方法，則會寫入的模擬 Amazon 日誌項目。

> **Note**
>
> 雖然 CloudWatch 日誌條目是模擬的，但方法調用的結果是真實的。

響應正文輸出應該是這樣的：

```
"Hello"
```

> **Tip**
>
> 嘗試使用不同的方法和無效的輸入來查看錯誤輸出的 API Gateway。您可能需要更改狀態機以查找特定密鑰，並在測試期間提供錯誤的密鑰以使狀態機執行失敗，並在響應體輸出中生成錯誤消息。

使用 cURL 測試已部署的 API

1. 開啟終端機視窗。
2. 複製以下 cURL 命令，並將它貼到終端機視窗，同時將 <api-id> 取代為您 API 的 API ID，以及將 <region> 取代為 API 的部署區域。

```
curl -X POST\
 'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
 -d '{"key":"Hello"}' \
 -H 'Content-Type: application/json'
```

響應體輸出應該是這樣的：

```
"Hello"
```

> **Tip**
>
> 嘗試使用不同的方法和無效的輸入來查看錯誤輸出的 API Gateway。您可能需要更改狀態機以查找特定密鑰，並在測試期間提供錯誤的密鑰以使狀態機執行失敗，並在響應主體輸出中生成錯誤消息。

步驟 4：清除

嘗試完 API Gateway 後，您可以使用 AWS CDK 拆除狀態機器和 API Gateway。在您的應用程式的主目錄中發行 `cdk destroy`。

AWS Step Functions 數據科學開發套件

資 AWS Step Functions 料科學 SDK 是供資料科學家使用的開放原始碼程式庫。使 SageMaker 用此 SDK，您可以建立使用和 Step Functions 來處理和發佈機器學習模型的工作流程。您也可以在 Python 中建立多步驟的機器學習工作流程，以大規模協調 AWS 基礎架構，而無需分別佈建和整合 AWS 服務。

資 AWS Step Functions 料科學 SDK 提供可建立和叫用 Step Functions 式工作流程的 Python API。您可以直接在 Python 以及 Jupyter 筆記本中管理和執行這些工作流程。

除了直接在 Python 中建立可供生產使用的工作流程外， AWS Step Functions Data Science SDK 還可讓您複製該工作流程、試驗新選項，然後將精細的工作流程投入生產環境中。

如需資 AWS Step Functions 料科學 SDK 的詳細資訊，請參閱下列內容：

- Github 上的專案
- 軟體開發套件文件
- 下列範例筆記本，可在SageMaker 主控台和相關GitHub 專案的 Jupyter 筆記本執行個體中使用：
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

使用地形部署狀態機

Terraform by HashiCorp 是使用基礎架構作為代碼（IaC）構建應用程序的框架。使用 Terraform，您可以建立狀態機器並使用功能，例如預覽基礎結構部署和建立可重複使用的範本。Terraform 範本可協助您維護和重複使用程式碼，方法是將程式碼分解成較小的區塊。

如果您熟悉 Terraform，可以遵循本主題中描述的開發生命週期，作為在 Terraform 中建立和部署狀態機器的模型。如果您不熟悉 Terraform，我們建議您先完成工作坊「地形簡介」，以了解地AWS形。

> **Tip**
>
> 要將使用 Terraform 構建的狀態機示例部署到您的AWS 帳戶，請參閱以基礎結構作為工作坊的代碼管理狀態機器模塊。AWS Step Functions

在本主題中

先決條件

開始之前，請確定您已完成下列先決條件：

- 在您的機器上安裝地形。如需有關安裝 Terraform 的資訊，請參閱安裝地形。
- 在計算機上安裝本地 Step Functions。我們建議您安裝 Step Functions 本地 Docker 映像以使用 Step Functions 本地。如需詳細資訊，請參閱在本地測試狀態機。
- 安裝 AWS SAM CLI。如需安裝資訊，請參閱AWS Serverless Application Model開發人員指南中的安裝 AWS SAM CLI。
- 安裝AWS Toolkit for Visual Studio Code以檢視狀態機器的工作流程圖。如需安裝資訊，請參閱《使用指南》AWS Toolkit for Visual Studio Code中的AWS Toolkit for Visual Studio Code〈安裝〉。

具有地形的狀態機開發生命週期

下列程序說明如何使用您在 Step Functions 式主控台中使用 Workflow Studio 建立的狀態機器原型，做為使用 Terraform 和. AWS Toolkit for Visual Studio Code

若要檢視討論使用 Terraform 進行狀態機器開發的完整範例，並詳細介紹最佳作法，請參閱撰寫 Step Functions Terraform 專案的最佳作法。

使用 Terraform 啟動狀態機的開發生命週期

1. 使用以下命令引導一個新的地形項目。

   ```
   terraform init
   ```

2. 開啟 Step Functions 主控台，為您的狀態機器建立原型。
3. 在工作流程 Studio 中，執行下列操作：

   a. 建立您的工作流程原型。

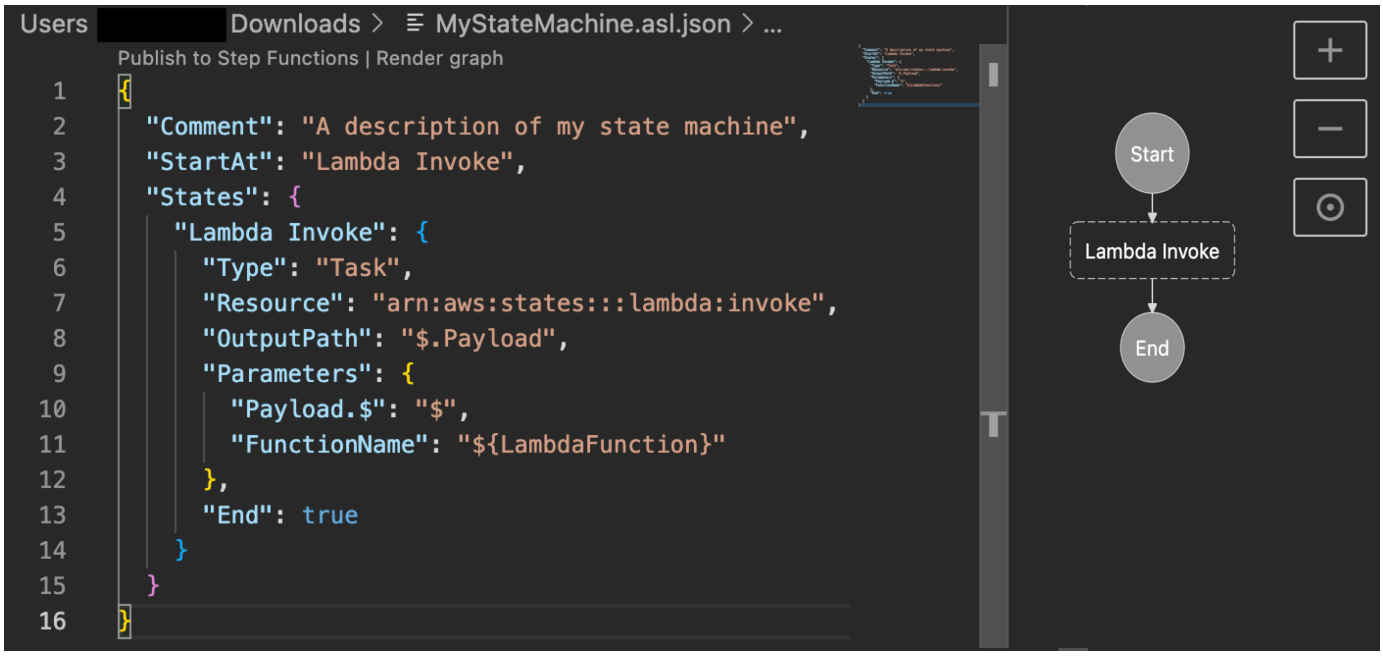
   b. 匯出工作流程的亞馬遜州語言 (ASL) 定義。若要這麼做，請選擇 [匯 Im port /Export/匯出] 下拉式清單，然後選取 [匯出 JSON 定義]。

4. 將匯出的 ASL 定義儲存在專案目錄中。

   您可以將匯出的 ASL 定義做為輸入參數傳遞至使用函數的 `aws_sfn_state_machine`Terraform 資源。`templatefile`此函數用於傳遞匯出的 ASL 定義和任何變數替代的定義欄位中。

   > **Tip**
   >
   > 由於 ASL 定義檔案可能包含冗長的文字區塊，因此建議您避免使用內嵌 EOF 方法。這樣可以更輕鬆地將參數替換為狀態機定義。

5. (選擇性) 更新 IDE 中的 ASL 定義，並使用視覺化您的變更。AWS Toolkit for Visual Studio Code



為了避免持續匯出定義並將其重構到專案中，我們建議您在 IDE 本機中進行更新，並使用 Git 追蹤這些更新。

6. 使用 Step Functions 本地測試您的工作流程。

> **Tip**
>
> 您也可以使用 AWS SAMCLI 本機在您的狀態機器中，在本機測試與 Lambda 函數和 API Gateway API 的服務整合。

7. 在部署狀態機之前預覽狀態機器和其他AWS資源。若要進行這項動作，請執行以下命令。

```
terraform plan
```

8. 使用下列指令，從本機環境或 C I/CD 管線部署狀態機器。

```
terraform apply
```

9. (選擇性) 使用下列命令清理資源並刪除狀態機器。

```
terraform destroy
```

狀態機器的 IAM 角色和政策

使用 Terraform 服務整合政策將必要的 IAM 許可新增至您的狀態機器，例如叫用 Lambda 函數的權限。您也可以定義明確的角色和原則，並將其與狀態機器相關聯。

下列 IAM 政策範例授予您的狀態機器存取權，以叫用名為的 Lambda 函數*myFunction*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
  ]
}
```

我們也建議您在 Terraform 中為狀態機器定義 IAM 政策時使用aws_iam_policy_document資料來源。這可協助您檢查政策是否格式錯誤，並以變數取代任何資源。

以下 IAM 政策範例使用aws_iam_policy_document資料來源並授予您的狀態機器存取權，以叫用名為的 Lambda 函數*myFunction*。

```
data "aws_iam_policy_document" "state_machine_role_policy" {

  statement {
    effect = "Allow"

    actions = [
      "lambda:InvokeFunction"
    ]

    resources = ["${aws_lambda_function.[[myFunction]].arn}:*"]
  }

}
```

> **Tip**
>
> 若要檢視使用 Terraform 部署的更多進階AWS架構模式，請參閱無伺服器土地工作流程集合中的 Terraform 範例。

測試與偵錯

Step Functions提供不同的方法來測試和偵錯狀態機器。例如，您可以在主控台中測試和偵錯狀態機器、使用 TestStateAPI 測試個別狀態，或使用 Local 在Step Functions本機測試狀態機器。

使用 TestStateAPI，您可以提供單一狀態的定義並執行它。您可以在不建立狀態機或更新現有狀態機的情況下測試單一狀態。

Step FunctionsLocal 是可下載的版本，Step Functions可讓您使用在自己的開發環境中Step Functions執行的版本來開發和測試應用程式。使用 L Step Functions ocal，您可以執行狀態機器，在本機開發環境中測試其輸入和輸出資料流程、與支援服務的整合等。

主題

- 使用 TestState API 來測試狀態
- 在本地測試狀態機

使用 TestState API 來測試狀態

TestStateAPI 接受單一狀態的定義並執行它。您可以在不建立狀態機或更新現有狀態機的情況下測試狀態。

使用 TestState API，您可以測試以下內容：

- 狀態的輸入和輸出處理數據流。
- AWS 服務 與其他 AWS 服務 請求和響應的集成
- 一個 HTTP 任務請求和響應

若要測試狀態，您也可以使用Step Functions主控台或 SDK。AWS Command Line Interface (AWS CLI)

`TestState`API 採用 IAM 角色，該角色必須包含狀態IAM存取資源的必要許可。如需狀態可能需要的權限的資訊，請參閱IAM使用 TestState API 的權限。

主題

- 關於使用 TestState API 的注意事項
- 在 TestState API 中使用檢驗層級
- IAM使用 TestState API 的權限

- 測試狀態（控制台）
- 使用測試狀態 AWS CLI
- 測試和調試輸入和輸出數據流

關於使用 TestState API 的注意事項

使用 TestStateAPI，您一次只能測試一個狀態。您可以測試的狀態包括下列各項：

- 所有工作類型，活動除外
- Pass
- 等候
- Choice
- Succeed
- Fail

使用 `TestState` API 時，請記住以下注意事項。

- 該 TestState API 不包括對以下內容的支持：
  - 任務使用下列資源類型的狀態：
    - 活動
    - 類型`.sync`或的服務整合模式 `.waitForTaskToken`
  - 平行狀態
  - Map狀態
- 一個測試最多可以運行五分鐘。如果測試超過此持續時間，則會失敗並顯示`States.Timeout`錯誤。

在 TestState API 中使用檢驗層級

若要使用 TestStateAPI 測試狀態，請提供該狀態的定義。然後測試返回一個輸出。對於每個狀態，您可以指定要在測試結果中檢視的詳細資訊量。這些詳細資料提供有關您正在測試的狀態的其他資訊。例如，如果您已使用任何輸入和輸出資料處理篩選器 (例如`InputPath`或`ResultPath`狀態)，則可以檢視中繼和最終資料處理結果。

Step Functions提供下列層次，以指定您要檢視的詳細資訊：

- 信息
- 除錯
- 追蹤

所有這些層級也會傳回status和nextState欄位。 status指示狀態執行的狀態。例如，SUCCEEDEDFAILED、RETRIABLE、和CAUGHT_ERROR。 nextState指示要轉換至的下一個狀態的名稱。如果您尚未在定義中定義下一個狀態，則此欄位會傳回空值。

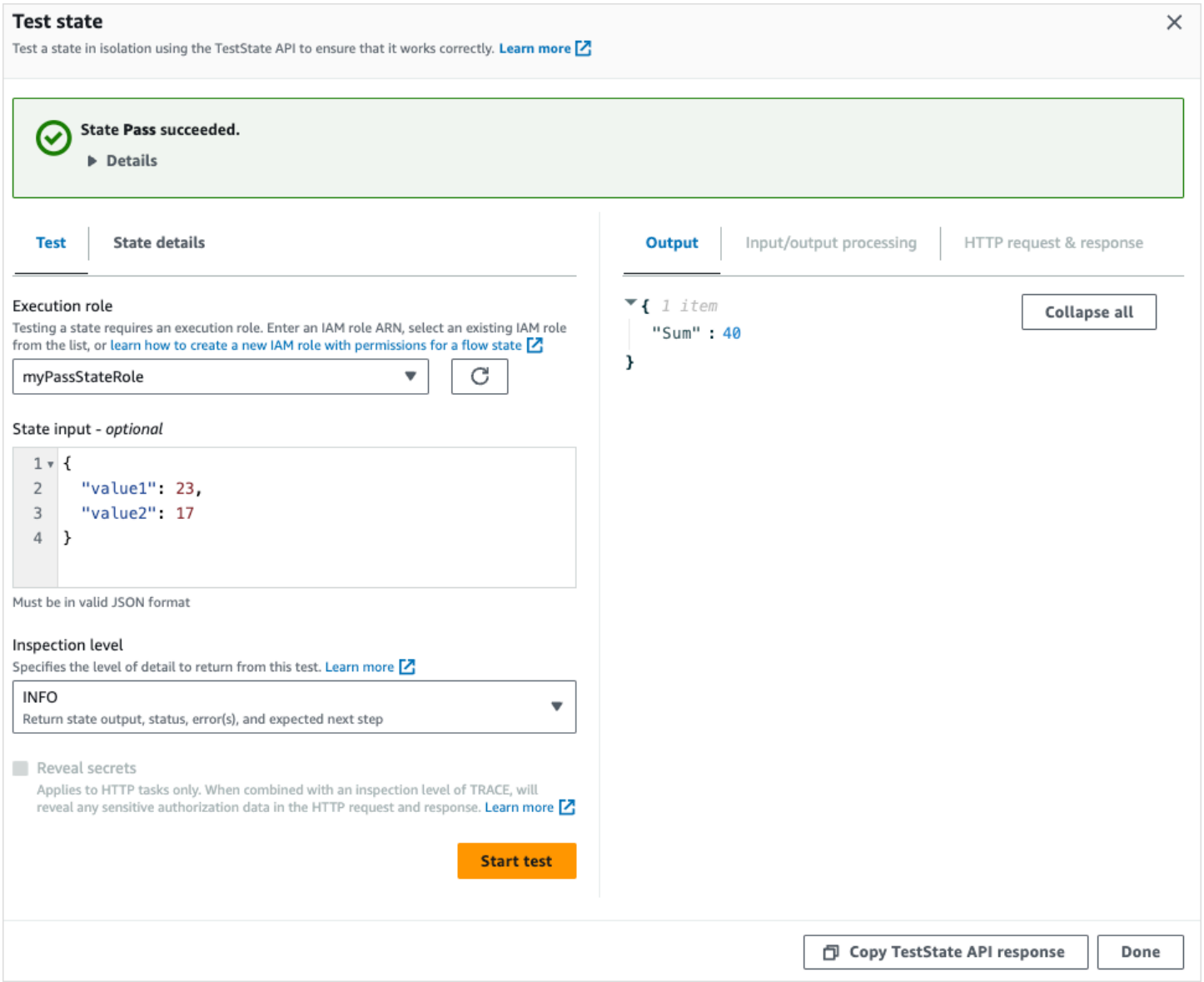
如需有關在Step Functions主控台中使用這些檢驗層級測試狀態的資訊AWS CLI，請參閱測試狀態（控制台）和使用測試狀態 AWS CLI。

資訊檢驗層級

如果測試成功，則此級別顯示狀態輸出。如果測試失敗，這個級別顯示錯誤輸出。依預Step Functions設，如果您未指定圖層，請將「檢驗層級」設定為 INFO。

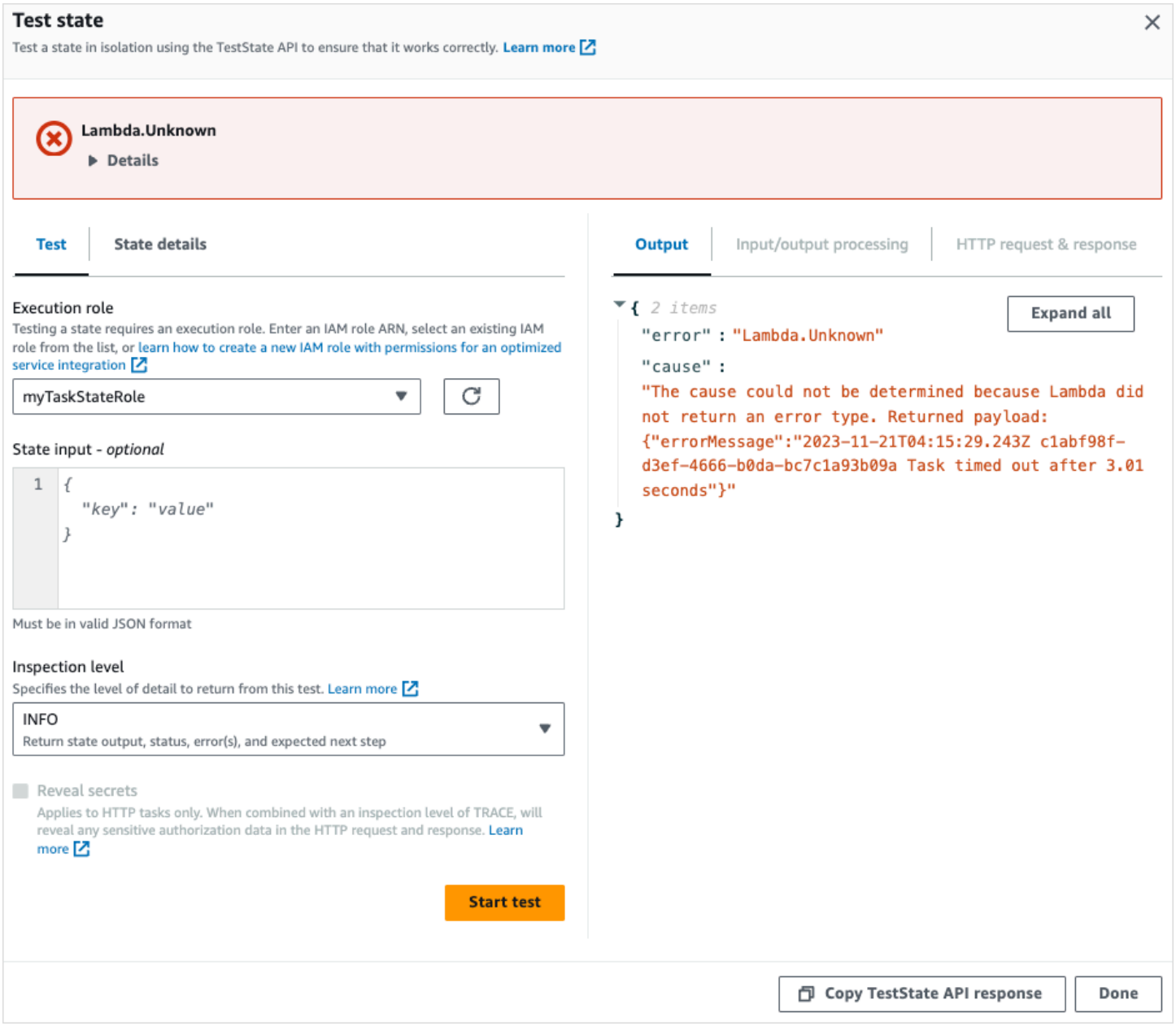
成功的 INFO 級別的測試示例

下圖顯示成功的「通過」狀態的測試。此狀態的 [檢驗層級] 設定為 [INFO]，且狀態的輸出會顯示在 [輸出] 索引標籤中。



具有 INFO 級別的測試示例

下圖顯示當 [檢驗層級] 設定為 INFO 時，[工作] 狀態的測試失敗。「輸出」索引標籤會顯示錯誤輸出，其中包含錯誤名稱以及該錯誤原因的詳細說明。



偵錯檢測層級

如果測試成功，則此級別顯示狀態輸出以及輸入和輸出數據處理的結果。

如果測試失敗，這個級別顯示錯誤輸出。此層級會顯示直到失敗點為止的中繼資料處理結果。例如，假設您測試了調用Lambda函數的 Task 狀態。假設您已將InputPath、參數ResultPath、和OutputPath篩選器套用至「工作」狀態。說調用失敗了。在此情況下，DEBUG層級會依照下列順序顯示根據篩選器應用程式的資料處理結果：

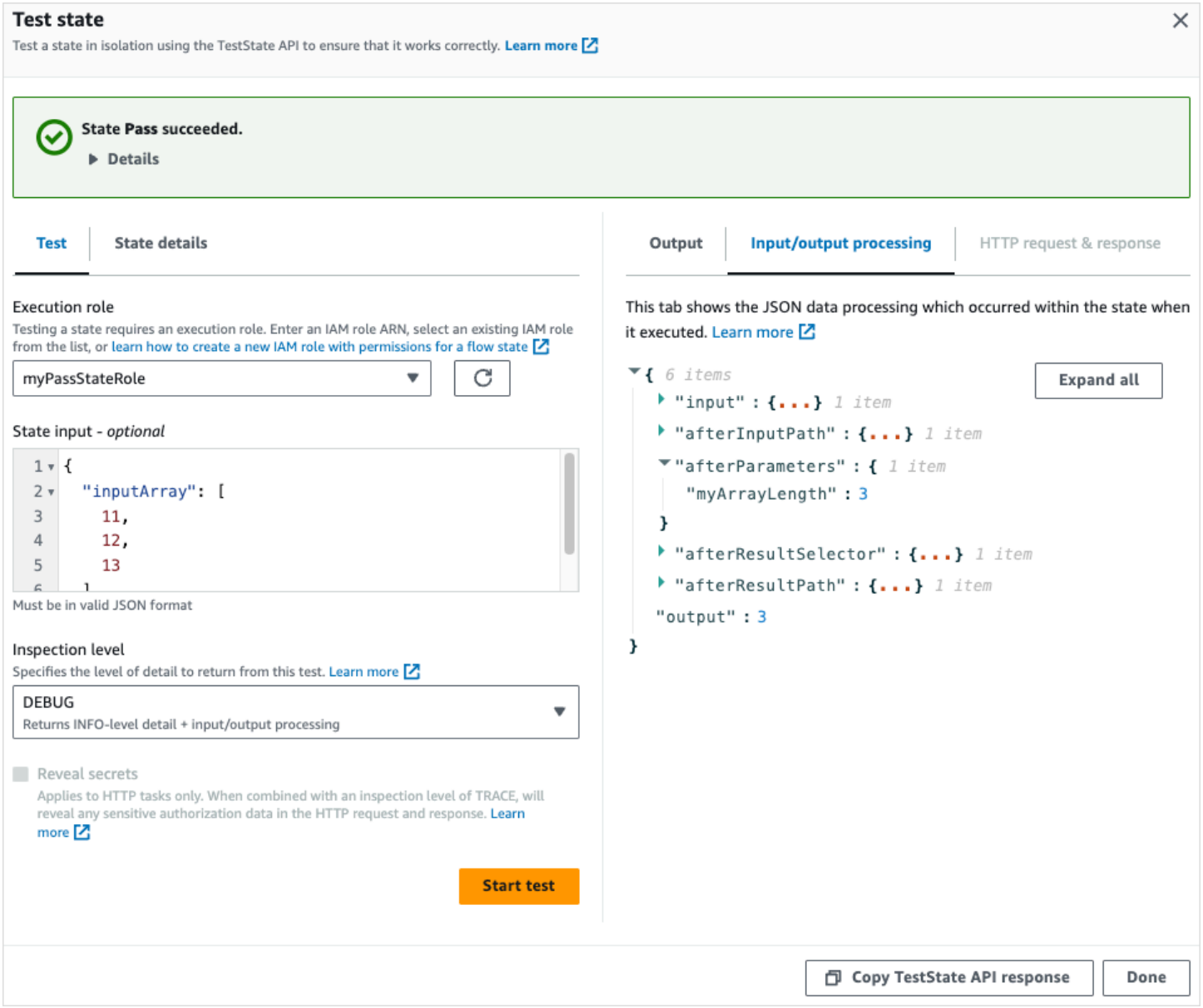
- `input`— 原始狀態輸入

- `afterInputPath`— Step Functions 應用`InputPath`過濾器後的輸入。
- `afterParameters`— Step Functions 應用`Parameters`過濾器後的有效輸入。

此層級中可用的診斷資訊可協助您疑難排解與您可能已定義的服務整合或輸入和輸出資料處理流程相關的問題。

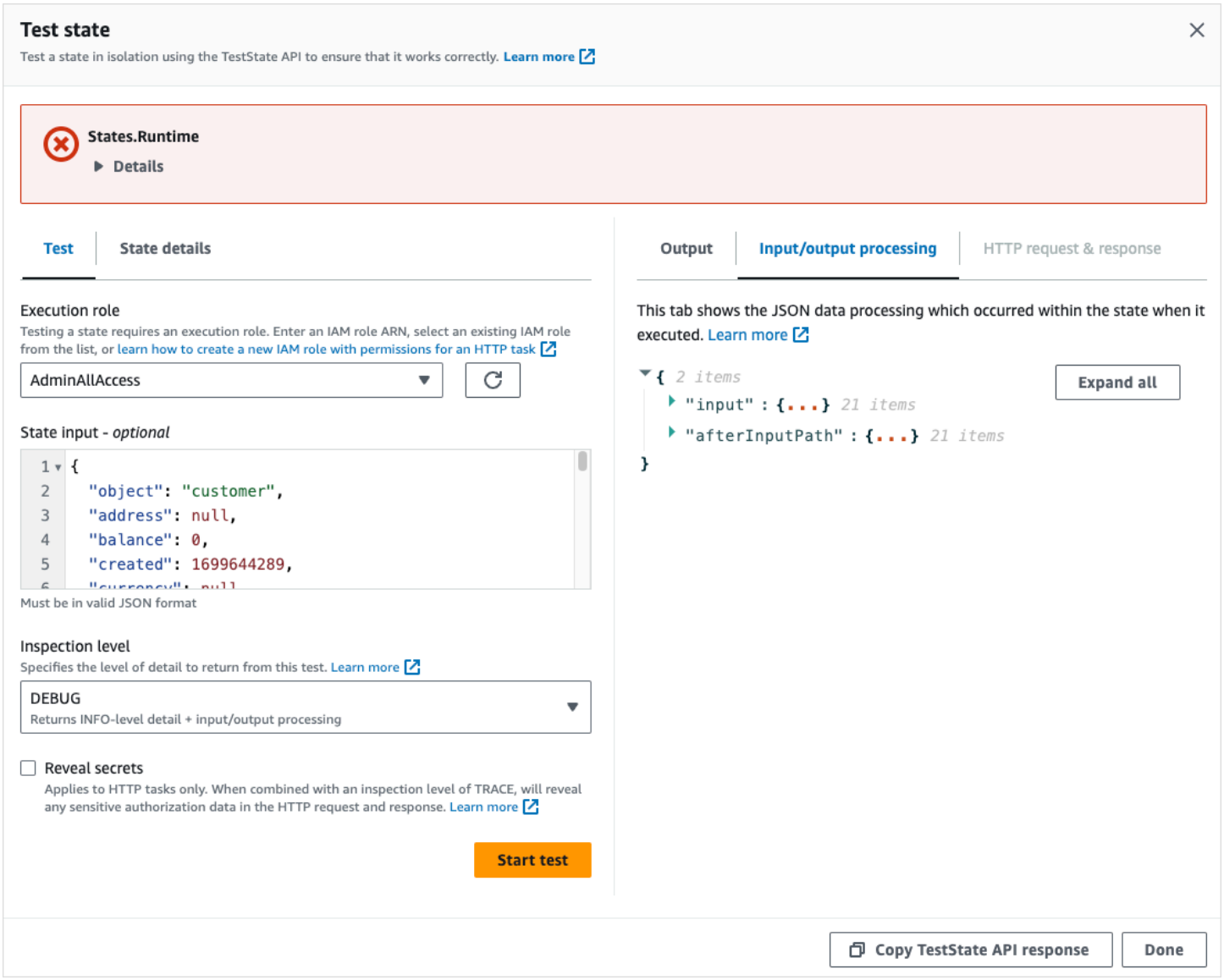
使用 DEBUG 級別成功的測試示例

下圖顯示成功的「通過」狀態的測試。此狀態的 [檢驗層級] 設定為 [偵錯]。下圖中的輸入/輸出處理選項卡顯示了針對此狀態提供的輸入`Parameters`上的應用程序的結果。



使用 DEBUG 級別進行測試的示例

下圖顯示當 [檢驗層級] 設定為 DE BUG 時，[工作] 狀態的測試失敗。下圖中的輸入/輸出處理選項卡顯示了直到其故障點的狀態的輸入和輸出數據處理結果。



追蹤檢測層級

Step Functions提供跟踪級別來測試 HTTP 任務。此層級會傳回第三方 API 傳回的 Step Functions HTTP 要求和回應的相關資訊。回應可能包含資訊，例如標頭和要求主體。此外，您可以檢視此層級中輸入和輸出資料處理的狀態輸出和結果。

如果測試失敗，這個級別顯示錯誤輸出。

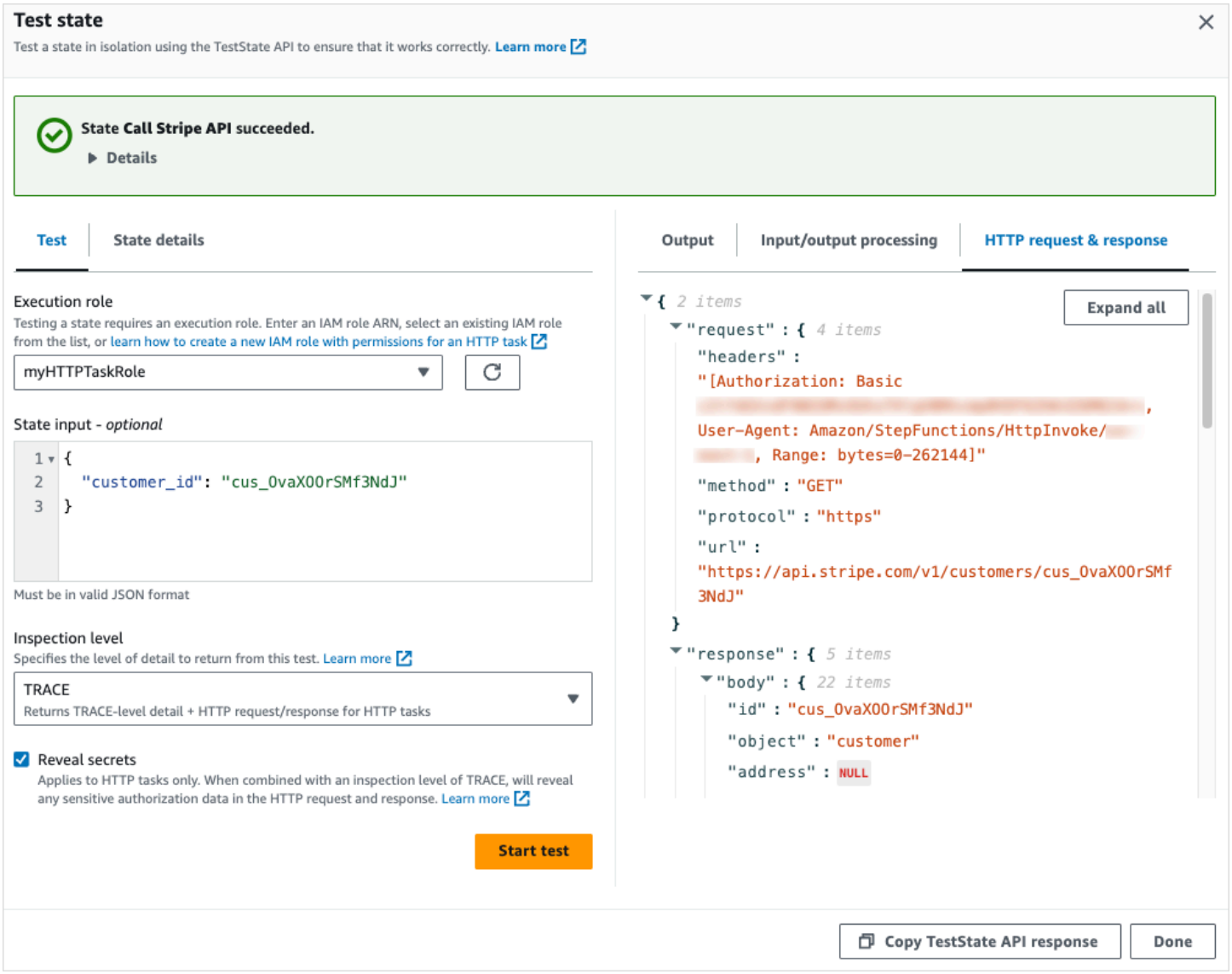
此層級僅適用於 HTTP 工作。 Step Functions如果您將此層級用於其他狀態類型，則會擲回錯誤。

當您將 [檢查] 層級設定為 TRACE 時，您也可以檢視EventBridge 連線中包含的密碼。若要這麼做，您必須在 TestStateAPI true 中將revealSecrets參數設定為。此外，您必須確定呼叫 TestState API 的IAM使用者具有states:RevealSecrets動作的權限。如需設定states:RevealSecrets權限的IAM原則範例，請參閱IAM使用 TestState API 的權限。如果沒有此權限，則Step Functions會擲回拒絕存取錯誤。

如果將revealSecrets參數設定為false，則會Step Functions忽略 HTTP 要求和回應資料中的所有密碼。

TRACE 級別成功的測試示例

下圖顯示成功之 HTTP 工作的測試。此狀態的 [檢驗層級] 設定為 [TRACE]。下圖中的 HTTP 要求與回應索引標籤會顯示第三方 API 呼叫的結果。



IAM使用 TestState API 的權限

呼叫 `TestState` API 的IAM使用者必須擁有執行`states:TestState`和`iam:PassRole`動作的權限。此外，如果您將 revealSecret 參數設定為`true`，則必須確定IAM使用者具有執行動作的權限。`states:RevealSecrets`如果沒有此權限，則Step Functions會擲回拒絕存取錯誤。

您還必須確保您的執行角色包含狀態正在訪問的資源所需的IAM權限。如需狀態可能需要之權限的相關資訊，請參閱管理執行角色。

下列IAM原則範例會設定`states:TestStateiam:PassRole`、和`states:RevealSecrets`權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

測試狀態（控制台）

您可以在控制台中測試狀態並檢查狀態輸出或輸入和輸出數據處理流程。對於 HTTP 任務，您可以測試原始 HTTP 請求和響應。

若要測試狀態

1. 開啟「Step Functions」主控台。
2. 選擇 [建立狀態機] 以開始建立狀態機，或選擇現有的狀態機。
3. 在工作流程工作室中，選擇您要測試設計模式的狀態。
4. 在工作流程工作室的Inspector面板中選擇測試狀態。
5. 在「測試狀態」對話方塊中，執行下列操作：

    a. 對於執行角色，請選擇要測試狀態的執行角色。請確定您具有要測試之狀態的必要IAM權限。

b. (選擇性) 提供測試所選狀態所需的任何 JSON 輸入。

c. 針對「檢驗層次」，根據您要檢視的值選取下列其中一個選項：

- INFO — 如果測試成功，則在「輸出」索引標籤中顯示狀態輸出。如果測試失敗，INFO 會顯示錯誤輸出，其中包括錯誤名稱和該錯誤原因的詳細說明。依預Step Functions設，如果您未選取圖層，請將「檢驗層級」設定為「資訊」。
- DEBUG — 如果測試成功，則顯示狀態輸出以及輸入和輸出資料處理的結果。如果測試失敗，DEBUG 會顯示錯誤輸出，其中包括錯誤名稱和該錯誤原因的詳細說明。
- TRACE — 顯示原始 HTTP 要求和回應，對於驗證標頭、查詢參數和其他 API 特定的詳細資訊非常有用。此選項僅適用於 HTTP 工作。

  或者，您可以選擇選取 [顯示密碼]。與 TRACE 結合使用時，此設定可讓您查看EventBridge連線插入的敏感資料，例如 API 金鑰。您用來存取主控台的使用IAM者身分必須具有執行`states:RevealSecrets`動作的權限。如果沒有此權限，則會在您啟動測試時Step Functions擲回拒絕存取錯誤。如需設定`states:RevealSecrets`權限的IAM原則範例，請參閱IAM使用 TestState API 的權限。

d. 選擇 [開始測試]。

使用測試狀態 AWS CLI

您可以使用中的 TestStateAPI 測試支援的狀態AWS CLI。此 API 接受狀態的定義並執行它。

對於每個狀態，您可以指定要在測試結果中檢視的詳細資訊量。這些詳細資料提供有關狀態執行的其他資訊，包括其輸入和輸出資料處理結果以及 HTTP 要求和回應資訊。下列範例展示您可以為 TestState API 指定的不同檢驗層級。請記住將`##`文本替換為特定於資源的信息。

本節包含下列範例，說明如何使用中Step Functions提供的不同檢驗層級AWS CLI：

- 使用資訊檢查層級
- 使用偵錯檢查層級
- 使用追蹤檢查層級
- 在中使用 jq 實用程序AWS CLI來過濾和打印 TestState API 返回的 HTTP 響應

範例 1：使用 INFO 檢查層級來測試「選擇」狀態

若要使用中的`INFO`檢查層級來測試狀態 AWS CLI，請執行`test-state`命令，如下列範例所示。

```
aws stepfunctions test-state \
    --definition '{"Type": "Choice", "Choices": [{"Variable": "$.number",
 "NumericEquals": 1, "Next": "Equals 1"}, {"Variable": "$.number", "NumericEquals": 2,
 "Next": "Equals 2"}], "Default": "No Match"}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --input '{"number": 2}'
```

此範例會使用 Choice 狀態，根據您提供的數值輸入來決定狀態的執行路徑。根據預Step Functions設，INFO如果您未設定樓層，則inspectionLevel將設定為。

Step Functions返回以下輸出。

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

範例 2：使用 DEBUG 檢查層級在「通過」狀態下偵錯輸入和輸出資料處理

若要使用中的DEBUG檢查層級來測試狀態 AWS CLI，請執行test-state命令，如下列範例所示。

```
aws stepfunctions test-state \
    --definition '{"Type": "Pass", "InputPath": "$.payload", "Parameters": {"data": 1},
 "ResultPath": "$.result", "OutputPath": "$.result.data", "Next": "Another State"}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --input '{"payload": {"foo": "bar"}}' \
    --inspection-level DEBUG
```

此範例使用Pass狀態來展示如何使用輸入和輸出資料處理Step Functions篩選器篩選和操作輸入 JSON 資料。此範例使用下列篩選器：InputPath##ResultPath、、和OutputPath。

Step Functions返回以下輸出。

```
{
    "output": "1",
    "inspectionData": {
        "input": "{\"payload\": {\"foo\": \"bar\"}}",
        "afterInputPath": "{\"foo\":\"bar\"}",
        "afterParameters": "{\"data\":1}",
        "afterResultSelector": "{\"data\":1}",
```

```
        "afterResultPath": "{\"payload\":{\"foo\":\"bar\"},\"result\":{\"data\":1}}"
    },
    "nextState": "Another State",
    "status": "SUCCEEDED"
}
```

範例 3：使用追蹤檢查層級和顯示機密來檢查傳送至協力廠商 API 的 HTTP 要求

若要使用TRACE檢查層級以及中的顯示秘密參數來測試 HTTP 工作，請執行test-state命令 AWS CLI，如下列範例所示。

```
aws stepfunctions test-state \
    --definition '{"Type": "Task", "Resource": "arn:aws:states:::http:invoke",
 "Parameters": {"Method": "GET", "Authentication": {"ConnectionArn":
 "arn:aws:events:us-
east-1:123456789012:connection/MyConnection/0000000-0000-0000-0000-000000000000"},
 "ApiEndpoint": "https://httpbin.org/get", "Headers": {"definitionHeader": "h1"},
 "RequestBody": {"message": "Hello from Step Functions!"}, "QueryParameters":
 {"queryParam": "q1"}}, "End": true}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --inspection-level TRACE \
    --reveal-secrets
```

此範例會測試 HTTP 工作是否呼叫指定的協力廠商 API、https://httpbin.org/。它也會顯示 API 呼叫的 HTTP 要求和回應資料。

```
{
    "output": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17 GMT\"],
\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],\"server\":
[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],\"content-
type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization
\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
    "inspectionData": {
        "input": "{}",
        "afterInputPath": "{}",
```

```
        "afterParameters": "{\"Method\":\"GET\",\"Authentication\":{\"ConnectionArn
\":\"arn:aws:events:us-east-1:123456789012:connection/foo/a59c10f0-a315-4c1f-
be6a-559b9a0c6250\"},\"ApiEndpoint\":\"https://httpbin.org/get\",\"Headers\":
{\"definitionHeader\":\"h1\"},\"RequestBody\":{\"message\":\"Hello from Step Functions!
\"},\"QueryParameters\":{\"queryParam\":\"q1\"}}",
        "result": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17 GMT\"],
\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],\"server\":
[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],\"content-
type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization
\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "afterResultSelector": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023
 00:06:17 GMT\"],\"access-control-allow-origin\":[\"*\"],\"content-length\":
[\"620\"],\"server\":[\"gunicorn/19.9.0\"],\"access-control-allow-credentials
\":[\"true\"],\"content-type\":[\"application/json\"]},\"ResponseBody\":{\"args
\":{\"QueryParam1\":\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":
{\"Authorization\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json;
 charset=UTF-8\",\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",
\"Host\":\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked
\",\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "afterResultPath": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17
 GMT\"],\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],
\"server\":[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],
\"content-type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization\":
\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "request": {
            "protocol": "https",
            "method": "GET",
```

```
            "url": "https://httpbin.org/get?
queryParam=q1&QueryParam1=QueryParamValue1",
            "headers": "[definitionHeader: h1, Authorization: Basic XXXXXXXX,
 CustomHeader1: CustomHeaderValue1, User-Agent: Amazon|StepFunctions|HttpInvoke|us-
east-1, Range: bytes=0-262144]",
            "body": "{\"message\":\"Hello from Step Functions!\",\"BodyKey1\":
\"BodyValue1\"}"
        },
        "response": {
            "protocol": "https",
            "statusCode": "200",
            "statusMessage": "OK",
            "headers": "[date: Tue, 21 Nov 2023 00:06:17 GMT, content-type:
 application/json, content-length: 620, server: gunicorn/19.9.0, access-control-allow-
origin: *, access-control-allow-credentials: true]",
            "body": "{\n  \"args\": {\n    \"QueryParam1\": \"QueryParamValue1\", \n
    \"queryParam\": \"q1\"\n  }, \n  \"headers\": {\n    \"Authorization\": \"Basic
 XXXXXXXX\", \n    \"Content-Type\": \"application/json; charset=UTF-8\", \n
 \"Customheader1\": \"CustomHeaderValue1\", \n    \"Definitionheader\": \"h1\", \n
   \"Host\": \"httpbin.org\", \n    \"Range\": \"bytes=0-262144\", \n    \"Transfer-
Encoding\": \"chunked\", \n    \"User-Agent\": \"Amazon|StepFunctions|HttpInvoke|us-
east-1\", \n    \"X-Amzn-Trace-Id\": \"Root=1-0000000-0000-0000-0000-000000000000\"\n
  }, \n  \"origin\": \"12.34.567.891\", \n  \"url\": \"https://httpbin.org/get?
queryParam=q1&QueryParam1=QueryParamValue1\"\n}\n"
        }
    },
    "status": "SUCCEEDED"
}
```

示例 4：使用 jq 實用程序過濾和打印 TestState API 返回的響應

該 TestState API 在響應中返回 JSON 數據作為轉義字符串。下列AWS CLI範例會擴充範例 3，並使用jq公用程式來篩選及列印 TestState API 以人類可讀格式傳回的 HTTP 回應。有關更多內容jq及其安裝說明，敬請參閱（詳見）。GitHub

```
aws stepfunctions test-state \
    --definition '{"Type": "Task", "Resource": "arn:aws:states:::http:invoke",
 "Parameters": {"Method": "GET", "Authentication": {"ConnectionArn":
 "arn:aws:events:us-
east-1:123456789012:connection/MyConnection/0000000-0000-0000-0000-000000000000"},
 "ApiEndpoint": "https://httpbin.org/get", "Headers": {"definitionHeader": "h1"},
 "RequestBody": {"message": "Hello from Step Functions!"}, "QueryParameters":
 {"queryParam": "q1"}}, "End": true}' \
```

```
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --inspection-level TRACE \
    --reveal-secrets \
    | jq '.inspectionData.response.body | fromjson'
```

下列範例會示範以人類可讀格式傳回的輸出。

```
{
  "args": {
    "QueryParam1": "QueryParamValue1",
    "queryParam": "q1"
  },
  "headers": {
    "Authorization": "Basic XXXXXXXX",
    "Content-Type": "application/json; charset=UTF-8",
    "Customheader1": "CustomHeaderValue1",
    "Definitionheader": "h1",
    "Host": "httpbin.org",
    "Range": "bytes=0-262144",
    "Transfer-Encoding": "chunked",
    "User-Agent": "Amazon|StepFunctions|HttpInvoke|us-east-1",
    "X-Amzn-Trace-Id": "Root=1-0000000-0000-0000-0000-000000000000"
  },
  "origin": "12.34.567.891",
  "url": "https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1"
}
```

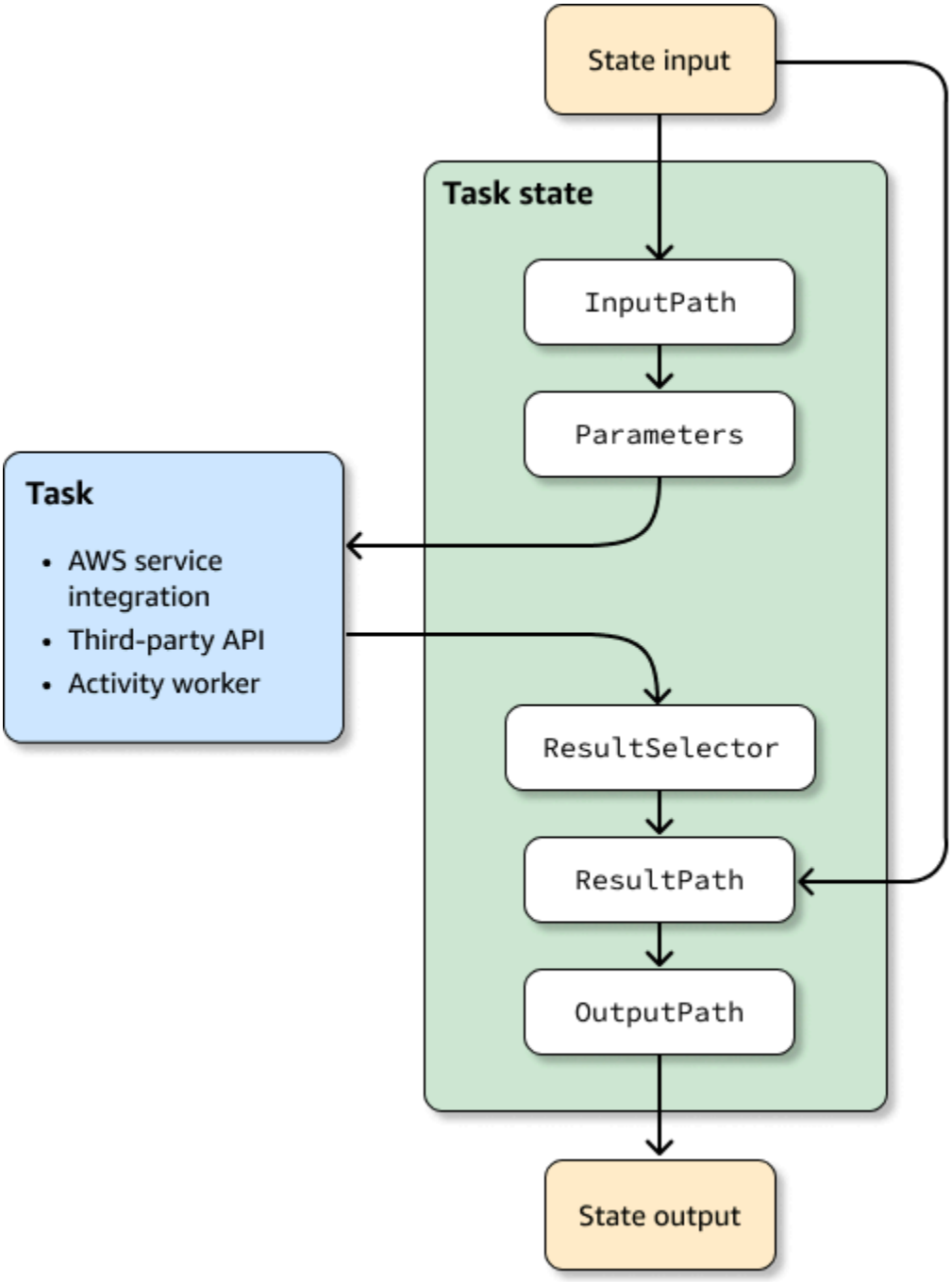
測試和調試輸入和輸出數據流

此 TestState API 有助於測試和偵錯流經工作流程的資料。本節提供一些重要概念，並說明如何使 TestState 用來達成此目的。

重要概念

在中Step Functions，在 JSON 資料通過狀態機器中的狀態時篩選和操作程序稱為輸入和輸出處理。如需其運作方式的相關資訊，請參閱Step Functions 中的輸入和輸出處理。

Amazon States Language(ASL) ([工作]、[平行]、[對應]、[暫不處理]、[等待]、[選擇]、[成功] 和 [失敗]) 中的所有狀態類型共用一組通用欄位，用於篩選和操作傳遞它們的 JSON 資料。這些欄位包括：InputPath參數ResultSelector、、ResultPath、和OutputPath。每個欄位的 Sup port 會因州而異。在執行階段，會以特定順序Step Functions套用每個欄位。下圖顯示將這些欄位套用至「工作」狀態內資料的順序：



下面的列表描述了圖中顯示的輸入和輸出處理字段的應用的順序。

1. 狀態輸入是從先前狀態傳遞到當前狀態的 JSON 數據。
2. InputPath過濾原始狀態輸入的一部分。
3. 參數配置要傳遞給任務的一組值。
4. 工作會執行工作並傳回結果。
5. ResultSelector從工作結果中選取一組要保留的值。

6. ResultPath結果與原始狀態輸入結合，或用它替換結果。
7. OutputPath篩選輸出的一部分，以傳遞至下一個狀態。
8. 狀態輸出是從當前狀態傳遞到下一個狀態的 JSON 數據。

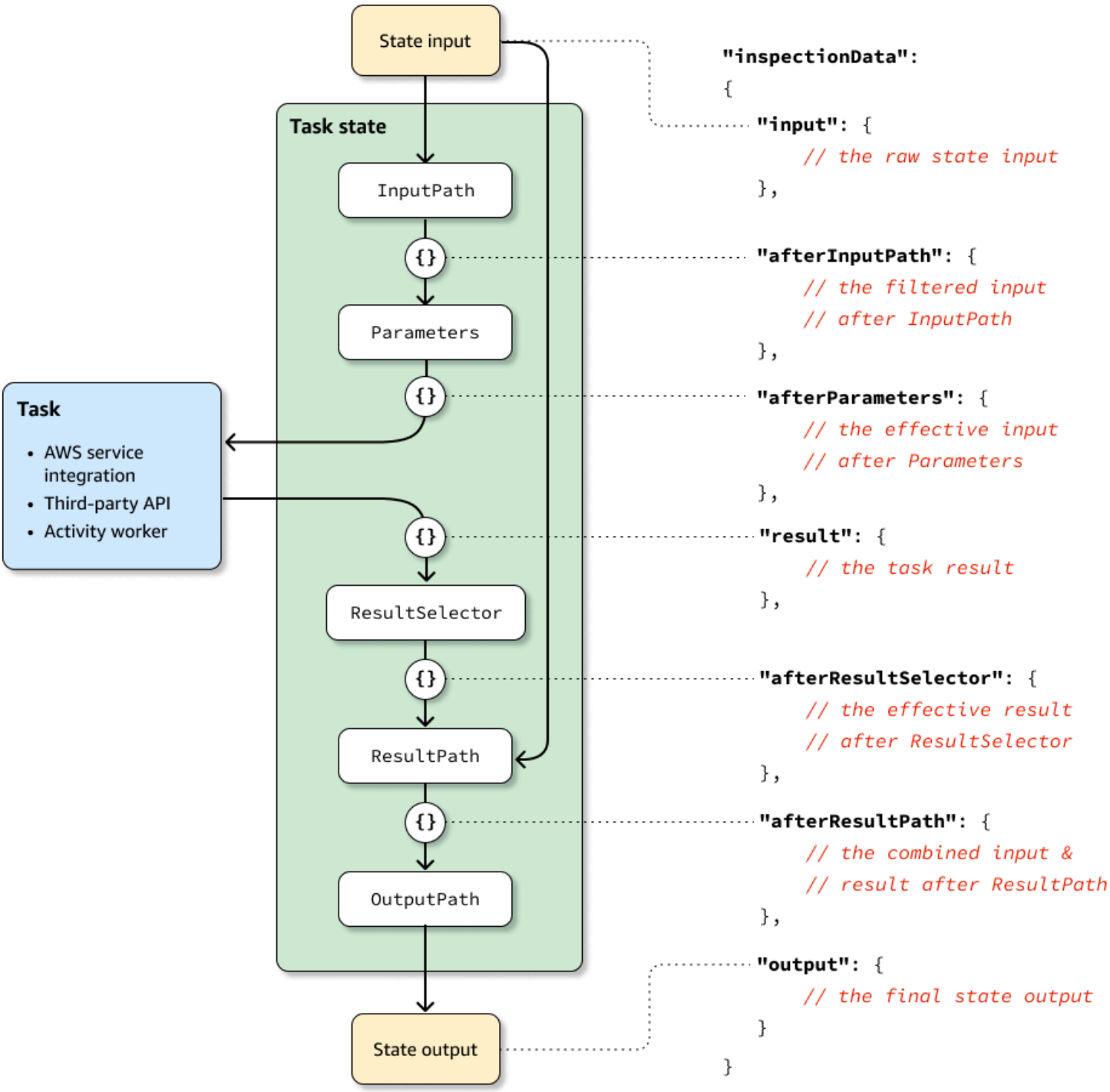
這些輸入和輸出處理字段是可選的。如果您在狀態定義中未使用任何這些欄位，工作會使用原始狀態輸入，並將工作結果傳回為狀態輸出。

用 TestState 於檢查輸入和輸出處理

當您呼叫 TestState API 並將inspectionLevel參數設定為時DEBUG，API 回應會包含名為的物件inspectionData。此物件包含欄位，可協助您檢查資料在執行時如何在狀態內篩選或操作。下列範例顯示「工作」狀態的inspectionData物件。

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

在此範例中，每個包含after前置詞的欄位都會在套用特定欄位後顯示資料。例如，afterInputPath顯示套用InputPath欄位來篩選原始狀態輸入的效果。下圖會將每個 ASL 定義欄位對應至inspectionData物件中對應的欄位：



如需使用 TestState API 偵錯輸入和輸出處理的範例，請參閱下列內容：

- 使用Step Functions控制台中的 DEBUG 檢查級別測試狀態
- 使用中的 DEBUG 檢測層級來測試狀態 AWS CLI

在本地測試狀態機

AWSStep Functions 本地是 Step Functions 的可下載版本，可讓您使用在自己的開發環境中運行的 Step Functions 版本開發和測試應用程序。Step Functions 的本機版本可以叫用AWS Lambda函式，無論是在本機執行AWS和執行時。您也可以協調其他支援的 AWS 服務。

> **Note**
>
> Step Functions 本地使用虛擬帳戶工作。

執行 Step Functions 本機時，您可以使用下列其中一種方式來叫用服務整合：

- 設定AWS Lambda和其他服務的本機端點。如需有關支援端點的資訊，請參閱設定 Step Functions 的組態選項本機。
- 撥打電話直接從 Step Functions 本地AWS服務。
- 嘲笑服務集成的響應。如需使用模擬服務整合的相關資訊，請參閱使用模擬服務集成。

AWSStep Functions 本地可作為一個 JAR 包或運行在 Microsoft 視窗，Linux，macOS 和其他平台，支持 Java 或碼頭的自包含碼頭圖像。

> **Warning**
>
> 的可下載版本AWS Step Functions僅用於測試，絕不應用於處理敏感資訊。

> **Tip**
>
> 確保您使用的是 Step Functions 本地版本 1.12.0 或更高版本，以便能夠在工作流程中包含所有內在函數。

下列主題說明如何使用 Docker 和 JAR 檔案設定 Step Functions 本機，以及如何執行 Step Functions 本機來使用AWS Lambda、AWS Serverless Application Model (AWS SAM) CLI 本機或其他支援的服務。

主題

- 設置 Step Functions 本地（可下載版本）和 Docker

- 設定步驟函數本機功能 (可下載版本)-Java 版本
- 設定 Step Functions 的組態選項本機
- 在您的計算機上運行 Step Functions 本地
- 測試步驟函數和 AWS SAM CLI 本地
- 使用模擬服務集成

設置 Step Functions 本地（可下載版本）和 Docker

Step Functions 本機 Docker 映像可讓您使用具有所有必要相依性的 Docker 映像，快速開始使用 Step Functions 本機。Docker 映像檔可讓您在容器化組建中包含 Step Functions 本機，並做為持續整合測試的一部分。

若要取得 Step Functions 本機的泊塢視窗影像，請參閱 https://hub.docker.com/r/amazon/ aws-stepfunctions-local，或輸入下列 Docker `pull` 命令。

```
docker pull amazon/aws-stepfunctions-local
```

要在 Docker 上啟動 Step Functions 的可下載版本，請運行以下 Docker 命令 `run`

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

若要與AWS Lambda其他支援的服務互動，您需要先設定認證和其他組態選項。如需詳細資訊，請參閱下列主題：

- 設定 Step Functions 的組態選項本機
- 泊塢視窗的認證和組態

設定步驟函數本機功能 (可下載版本)-Java 版本

的可下載版本AWS Step Functions是以可執行 JAR 檔案和 Docker 影像的形式提供。Java 應用程式可在 Windows、Linux、macOS 和其他支援 Java 的平台上執行。除了 Java 之外，您還需要安裝 AWS Command Line Interface (AWS CLI)。若要取得有關安裝和配置的資訊AWS CLI，請參閱《AWS Command Line Interface使用指南》。

在電腦上設定和執行 Step Functions

1. 使用以下鏈接下載 Step Functions。

下載連結	檢查總和
.tar.gz	.tar.gz.md5
.zip	. 拉鍊.

2. 解壓縮 `.zip` 檔案。
3. 測試下載及檢視版本資訊。

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (選用) 檢視可用命令的清單。

```
$ java -jar StepFunctionsLocal.jar -h
```

5. 若要在電腦上啟動 Step Functions，請開啟命令提示字元，導覽至您解壓縮的目錄`StepFunctionsLocal.jar`，然後輸入下列命令。

```
java -jar StepFunctionsLocal.jar
```

6. 若要存取在本機執行的 Step Functions 數，請使用`--endpoint-url`參數。例如，使用AWS CLI，您可以指定「Step Functions」指令，如下所示：

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> 依預設，Step Functions 本機使用本機測試帳戶和認證，而且AWS區域設定為美國東部 (維吉尼亞北部)。若要使用 Step Functions 本機搭AWS Lambda配使用或其他支援的服務，您必須設定認證和區域。
> 如果您將 Express 工作流程與 Step Functions 本機搭配使用，則執行歷程記錄將儲存在記錄檔中。它不會記錄到 CloudWatch 記錄檔。記錄檔路徑將以建立本機狀態機器時提供的 CloudWatch 記錄記錄群組 ARN 為基礎。記錄檔將儲存在`/aws/states/log-group-`

name/*${execution_arn}*.log相對於您執行 Step Functions 本機的位置。例如，如果執行 ARN 是：

```
arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI
```

日誌檔案將是：

```
aws/states/log-group-name/arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
```

設定 Step Functions 的組態選項本機

當您使用 JAR 檔案啟動「AWS Step Functions本機」時，您可以使用 AWS Command Line Interface (AWS CLI) 或將它們包含在系統環境中來設定組態選項。對於 Docker，您必須在啟動 Step Functions 本機時參考的檔案中指定這些選項。

組態選項

當您將 Step Functions 本機容器設定為使用覆寫端點 (例如 Lambda 端點和 Batch 端點)，並呼叫該端點時，Step Functions 本機不會使用您指定的認證。設定這些端點覆寫是選擇性的。

選項	命令列	Environment
帳戶	-帳戶，-AW-帳戶	AWS_ACCOUNT_ID
區域	-區域，-aws-region	AWS_DEFAULT_REGION
等待時間調整	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Lambda 端點	-Lambda 端點，-Lambda 端點	LAMBDA_ENDPOINT
批次端點	-批處理端點，批處理端點	BATCH_ENDPOINT
DynamoDB 端點	-動態端點，-動態端點	DYNAMODB_ENDPOINT
ECS 端點	-回收點，-EC-端點	ECS_ENDPOINT

選項	命令列	Environment
Glue 端點	-膠端點，膠水端點	GLUE_ENDPOINT
SageMaker 端點	-sageMakerEndpoint，-SAGEMAKER 端點	SAGE_MAKER_ENDPOINT
SQS 端點	-SQS-端點，-sqs 端點	SQS_ENDPOINT
SNS 端點	-訊號點，-sns 端點	SNS_ENDPOINT
Step Functions 端點	-stepFunctionsEndpoint, --step-functions-endpoint	步驟函數端點

泊塢視窗的認證和組態

若要設定 Docker 的本機 Step Functions，請建立下列檔案：`aws-stepfunctions-local-credentials.txt`。

此檔案包含您的認證和其他組態選項。創建`aws-stepfunctions-local-credentials.txt`文件時，可以將以下內容用作模板。

```
AWS_DEFAULT_REGION=AWS_REGION_OF_YOUR_AWS_RESOURCES
AWS_ACCESS_KEY_ID=YOUR_AWS_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=YOUR_AWS_SECRET_KEY
WAIT_TIME_SCALE=VALUE
LAMBDA_ENDPOINT=VALUE
BATCH_ENDPOINT=VALUE
DYNAMODB_ENDPOINT=VALUE
ECS_ENDPOINT=VALUE
GLUE_ENDPOINT=VALUE
SAGE_MAKER_ENDPOINT=VALUE
SQS_ENDPOINT=VALUE
SNS_ENDPOINT=VALUE
STEP_FUNCTIONS_ENDPOINT=VALUE
```

在中設定認證和組態選項之後`aws-stepfunctions-local-credentials.txt`，請使用下列命令啟動 Step Functions。

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> 建議使用特殊的 DNS 名稱`host.docker.internal`，該名稱會解析為主機使用的內部 IP 位址，例如`http://host.docker.internal:8000`。如需詳細資訊，請參閱適用於 Mac 和 Windows 的泊塢視窗文件，以及 Mac 版泊塢視窗桌面中的網路功能和 Windows 版泊塢視窗桌面中的網路功能。

在您的計算機上運行 Step Functions 本地

使用本機版本的 Step Functions 來設定、開發和測試電腦上的狀態機器。

在本地運行 HelloWorld 狀態機

使用 AWS Command Line Interface (AWS CLI) 在本機執行 Step Functions 式之後，您可以啟動狀態機器執行。

1. AWS CLI藉由逸出狀態機定義，從中建立狀態機器。

```
aws stepfunctions --endpoint-url http://localhost:8083 create-state-machine --
definition "{\
  \"Comment\": \"A Hello World example of the Amazon States Language using a Pass
 state\",\
  \"StartAt\": \"HelloWorld\",\
  \"States\": {\
    \"HelloWorld\": {\
      \"Type\": \"Pass\",\
      \"End\": true\
    }\
  }}" --name "HelloWorld" --role-arn "arn:aws:iam::012345678901:role/DummyRole"
```

> **Note**
>
> 不用於 Step Functions 局部，但您必須使用適當的語法來包含它。`role-arn`您可以使用上一個範例的 Amazon Resource Name (ARN)。

如果您成功建立狀態機，Step Functions 會回應建立日期和狀態機 ARN。

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. 使用您建立的狀態機器 ARN 開始執行。

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions 本地使用 AWS SAM CLI 本地

您可以將本機版本的 Step Functions 與本機版本搭配使用AWS Lambda。如要進行設定，您必須安裝和設定 AWS SAM。

如需設定和執行 AWS SAM 的資訊，請參閱以下內容：

- 設定 AWS SAM
- 啟動 AWS SAM CLI Local。

當 Lambda 在本機系統上執行時，您可以啟動本機 Step Functions。從您擷取 Step Functions 數本機 JAR 檔案的目錄中，啟動本機 Step Functions 數，然後使用`--lambda-endpoint`參數來設定本機 Lambda 端點。

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

如需使用本機執行 Step Functions 的詳細資訊AWS Lambda，請參閱測試步驟函數和 AWS SAM CLI 本地。

測試步驟函數和 AWS SAM CLI 本地

同時AWS Step Functions並在本機電腦上AWS Lambda執行時，您可以測試狀態機器和 Lambda 函數，而無需將程式碼部署到AWS。

如需詳細資訊，請參閱下列主題：

- 在本地測試狀態機
- 設定 AWS SAM

主題

步驟一：設定 AWS SAM

AWS Serverless Application Model (AWS SAM) CLI Local 需要安裝 AWS Command Line Interface、AWS SAM 和 Docker。

1. 安裝 AWS SAM CLI。

   > **Note**
   >
   > 在安裝 AWS SAM CLI 前，您需要安裝 AWS CLI 和 Docker。請參閱安裝 AWS SAM CLI 的先決條件。

2. 請參閱 AWS SAM Quick Start 文件。請務必遵循步驟來執行以下作業：

   1. 初始化應用程式
   2. 在本機測試應用程式

   這將創建一個`sam-app`目錄，並構建一個包含基於 Python 的 Hello World Lambda 函數的環境。

步驟 2：測試 AWS SAM CLI Local

現在您已經安裝AWS SAM並建立了 Hello World Lambda 函數，您可以測試該函數。在 `sam-app` 目錄中，輸入下列命令：

```
sam local start-api
```

這會啟動 Lambda 函數的本機執行個體。您應該會看到類似下列內容的輸出：

```
2019-01-31 16:40:27 Found credentials in shared credentials file: ~/.aws/credentials
2019-01-31 16:40:27 Mounting HelloWorldFunction at http://127.0.0.1:3000/hello [GET]
2019-01-31 16:40:27 You can now browse to the above endpoints to invoke your functions.
 You do not need to restart/reload SAM CLI while working on your functions changes will
 be reflected instantly/automatically. You only need to restart SAM CLI if you update
 your AWS SAM template
2019-01-31 16:40:27  * Running on http://127.0.0.1:3000/ (Press CTRL+C to quit)
```

開啟瀏覽器並輸入下列內容：

```
http://127.0.0.1:3000/hello
```

這將輸出類似於以下內容的響應：

```
{"message": "hello world", "location": "72.21.198.66"}
```

輸入 CTRL+C 以結束 Lambda API。

步驟 3：啟動 AWS SAM CLI Local

現在您已測試該函數可正常運作，請啟動 AWS SAM CLI Local。在 `sam-app` 目錄中，輸入下列命令：

```
sam local start-lambda
```

這會啟動 AWS SAM CLI 本機，並提供要使用的端點，類似下列輸出：

```
2019-01-29 15:33:32 Found credentials in shared credentials file: ~/.aws/credentials
2019-01-29 15:33:32 Starting the Local Lambda Service. You can now invoke your Lambda
 Functions defined in your template through the endpoint.
2019-01-29 15:33:32  * Running on http://127.0.0.1:3001/ (Press CTRL+C to quit)
```

第 4 步：啟動步驟函數本地

JAR 檔案

如果您使用的是本機步驟函數的`.jar`檔案版本，請啟動步驟函數並指定 Lambda 端點。在解壓縮`.jar`檔案的目錄中，輸入下列命令：

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

當步驟函數本機啟動時，它會檢查環境，然後檢查您`~/.aws/credentials`檔案中設定的認證。默認情況下，它開始使用虛擬的用戶 ID，並列為。`region us-east-1`

```
2019-01-29 15:38:06.324: Failed to load credentials from environment because Unable to
 load AWS credentials from environment variables (AWS_ACCESS_KEY_ID (or AWS_ACCESS_KEY)
 and AWS_SECRET_KEY (or AWS_SECRET_ACCESS_KEY))
2019-01-29 15:38:06.326: Loaded credentials from profile: default
2019-01-29 15:38:06.326: Starting server on port 8083 with account 123456789012, region
 us-east-1
```

Docker

如果您使用的是 Docker 版本的步驟函數本機，請使用下列命令啟動步驟函數：

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

如需有關安裝 Docker 版本的步驟函式的資訊，請參閱設置 Step Functions 本地（可下載版本）和 Docker。

> **Note**
>
> 您可以透過指令行指定端點，或在從`.jar`檔案啟動 Step Functions 時設定環境變數來指定端點。針對 Docker 版本，您必須在文字檔案中指定端點和登入資料。請參閱 設定 Step Functions 的組態選項本機。

步驟 5：建立參考您 AWS SAM CLI Local 函數的狀態機器

一旦步驟函數本地運行，創建一個引用您初始化的狀態機步驟一：設定 AWS SAM。`HelloWorldFunction`

```
aws stepfunctions --endpoint http://localhost:8083 create-state-machine --definition
 "{\
  \"Comment\": \"A Hello World example of the Amazon States Language using an AWS
 Lambda Local function\",\
  \"StartAt\": \"HelloWorld\",\
  \"States\": {\
    \"HelloWorld\": {\
      \"Type\": \"Task\",\
      \"Resource\": \"arn:aws:lambda:us-east-1:123456789012:function:HelloWorldFunction
\",\
      \"End\": true\
    }\
  }\
}\
}}" --name "HelloWorld" --role-arn "arn:aws:iam::012345678901:role/DummyRole"
```

這將創建一個狀態機，並提供一個亞馬遜資源名稱（ARN），您可以用它來啟動執行。

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

步驟 6：開始您本機狀態機器的執行

一旦你創建了一個狀態機，開始執行。使用以下**aws stepfunctions**命令時，您需要引用端點和狀態機 ARN：

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

這將啟動您的HelloWorld狀態機器命test名的執行。

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

現在，步驟函數在本機執行，您可以使用AWS CLI. 例如，若要取得有關此執行的資訊，請使用下列命令：

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

呼叫執`describe-execution`行會提供更完整的詳細資訊，類似下列輸出：

```
{
    "status": "SUCCEEDED",
    "startDate": 1549056334.073,
    "name": "test",
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test",
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
    "stopDate": 1549056351.276,
    "output": "{\"statusCode\": 200, \"body\": \"{\\\"message\\\": \\\"hello world\\\",
 \\\"location\\\": \\\"72.21.198.64\\\"}\"}",
    "input": "{}"
}
```

使用模擬服務集成

在 Step Functions Local 中，您可以使用模擬服務整合來測試狀態機器的執行路徑，而無需實際呼叫整合式服務。要將狀態機器配置為使用模擬服務集成，請創建一個模擬配置文件。在此文件中，您將服務集成的所需輸出定義為模擬響應，以及使用模擬響應將執行路徑模擬為測試用例的執行的執行。

藉由將模擬設定檔提供給 Step Functions Local，您可以執行使用測試案例中指定的模擬回應的狀態機器來測試服務整合呼叫，而不是進行實際的服務整合呼叫。

> **Note**
>
> 如果您沒有在模擬配置文件中指定模擬服務集成響應，Step Functions 本地將使用您在設置步 Step Functions 本地配置的端點調用 AWS 服務集成。如需有關設定 Step Functions 本機端點的資訊，請參閱設定 Step Functions 的組態選項本機。

主題

- 本主題的關鍵概念
- 步驟 1：在模擬配置文件中指定模擬服務集成
- 第 2 步：提供模擬配置文件 Step Functions 本地

- 步驟 3：執行模擬服務整合測試
- 模擬服務整合的組態檔

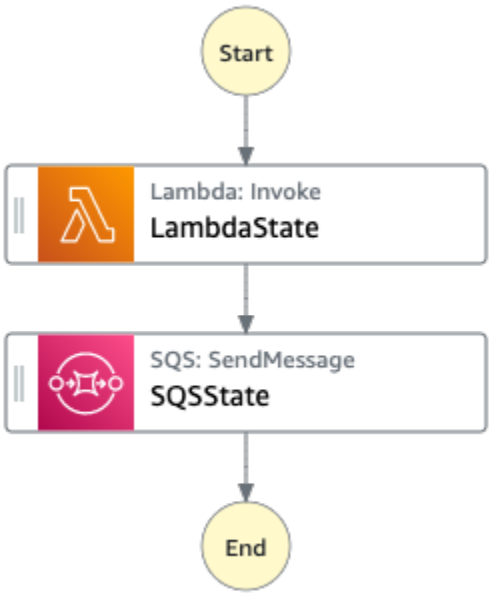
本主題的關鍵概念

本主題使用下列清單中定義的數個概念：

- 模擬服務集成-指配置為使用模擬響應而不是執行實際服務調用的任務狀態。
- 模擬響應-指任務狀態可以配置為使用的模擬數據。
- 測試用例-是指配置為使用模擬服務集成的狀態機器執行。
- 模擬配置文件-指包含 JSON 的模擬配置文件，該文件定義了模擬服務集成，模擬響應和測試用例。

步驟 1：在模擬配置文件中指定模擬服務集成

您可以使用 Step Functions 本機來測試 Step Functions AWS SDK 和最佳化服務整合。下圖顯示了狀態機器定義標籤中定義的狀態機器：



要做到這一點，你必須創建一個模擬配置文件，其中包含如中所定義的部分引入模擬配置的結構。

1. 建立名為的檔案MockConfigFile.json以設定具有模擬服務整合的測試。

   下列範例會示範參照狀態機器的模擬組態檔案，其中兩個已定義的狀態為LambdaState和SQSState。

Mock configuration file example

以下是模擬組態檔案的範例，該檔案示範如何從叫用 Lambda 函數並將訊息傳送至 Amazon SQS 來模擬回應。在此範例中，LambdaSQSIntegration狀態機器包含三個名為HappyPath、的測試案例RetryPath，以及HybridPath模擬命名為LambdaState和的Task狀態SQSState。這些狀態會使用MockedLambdaSuccessMockedSQSSuccess、和MockedLambdaRetry模擬服務回應。這些模擬服務響應在文件的MockedResponses部分中定義。

```
{
  "StateMachines":{
    "LambdaSQSIntegration":{
      "TestCases":{
        "HappyPath":{
          "LambdaState":"MockedLambdaSuccess",
          "SQSState":"MockedSQSSuccess"
        },
        "RetryPath":{
          "LambdaState":"MockedLambdaRetry",
          "SQSState":"MockedSQSSuccess"
        },
        "HybridPath":{
          "LambdaState":"MockedLambdaSuccess"
        }
      }
    }
  },
  "MockedResponses":{
    "MockedLambdaSuccess":{
      "0":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
          }
        }
      }
    },
    "LambdaMockedResourceNotReady":{
      "0":{
        "Throw":{
```

```
          "Error":"Lambda.ResourceNotReadyException",
          "Cause":"Lambda resource is not ready."
        }
      }
    },
    "MockedSQSSuccess":{
      "0":{
        "Return":{
          "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
          "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
        }
      }
    },
    "MockedLambdaRetry":{
      "0":{
        "Throw":{
          "Error":"Lambda.ResourceNotReadyException",
          "Cause":"Lambda resource is not ready."
        }
      },
      "1-2":{
        "Throw":{
          "Error":"Lambda.TimeoutException",
          "Cause":"Lambda timed out."
        }
      },
      "3":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
          }
        }
      }
    }
  }
}
```

State machine definition

以下是一個名為的狀態機器定義的範例LambdaSQSIntegration，它定義了兩個名為LambdaState和的服務整合工作狀態SQSState。 LambdaState包含以下項目的重試策略States.ALL。

```
{
  "Comment":"This state machine is called: LambdaSQSIntegration",
  "StartAt":"LambdaState",
  "States":{
    "LambdaState":{
      "Type":"Task",
      "Resource":"arn:aws:states:::lambda:invoke",
      "Parameters":{
        "Payload.$":"$",
        "FunctionName":"HelloWorldFunction"
      },
      "Retry":[
        {
          "ErrorEquals":[
            "States.ALL"
          ],
          "IntervalSeconds":2,
          "MaxAttempts":3,
          "BackoffRate":2
        }
      ],
      "Next":"SQSState"
    },
    "SQSState":{
      "Type":"Task",
      "Resource":"arn:aws:states:::sqs:sendMessage",
      "Parameters":{
        "QueueUrl":"https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
        "MessageBody.$":"$"
      },
      "End": true
    }
  }
}
```

您可以使用以下測試用例之一運行模擬配置文件中引用的LambdaSQSIntegration狀態機定義：

- HappyPath-此測試MockedSQSSuccess分別嘲笑LambdaState和SQSState使用MockedLambdaSuccess和的輸出。
  - LambdaState將返回以下值：

```
"0":{
  "Return":{
    "StatusCode":200,
    "Payload":{
      "StatusCode":200,
      "body":"Hello from Lambda!"
    }
  }
}
```

  - SQSState將返回以下值：

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- RetryPath-此測試MockedSQSSuccess分別嘲笑LambdaState和SQSState使用MockedLambdaRetry和的輸出。此外，還設定LambdaState為執行四次重試嘗試。這些嘗試的模擬回應會在狀態中定義並建MockedLambdaRetry立索引。
  - 初始嘗試以工作失敗結束，其中包含原因和錯誤訊息，如下列範例所示：

```
"0":{
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
}
```

  - 第一次和第二次重試嘗試以工作失敗結束，其中包含原因和錯誤訊息，如下列範例所示：

```
"1-2":{
```

```
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

- 第三次重試嘗試以任務成功結束，其中包含來自模擬 Lambda 回應中「有效負載」區段的狀態結果。

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
      "body": "Hello from Lambda!"
    }
  }
}
```

> **Note**
>
> - 對於具有重試原則的狀態，「Step Functions 本機」會耗盡原則中設定的重試嘗試，直到收到成功回應為止。這意味著您必須表示具有連續嘗試次數的重試模擬，並且應該在返回成功響應之前覆蓋所有重試嘗試。
> - 如果您沒有為特定的重試嘗試指定模擬回應 (例如重試「3」)，則狀態機器執行將會失敗。

- HybridPath-此測試嘲笑的LambdaState輸出。成功LambdaState運行並接收模擬數據作為響應後，對生產中指定的資源SQSState執行實際的服務調用。

有關如何使用模擬服務集成開始測試執行的信息，請參閱。步驟 3：執行模擬服務整合測試

2. 請確定模擬回應的結構符合您進行整合式服務呼叫時所收到的實際服務回應結構。如需有關模擬回應之結構需求的資訊，請參閱設定模擬服務整合。

   在前面的示例模擬配置文件中，在中定義的模擬響應MockedLambdaSuccess並MockedLambdaRetry符合從調用HelloFromLambda返回的實際響應的結構。

> **Important**
>
> AWS 不同服務之間的服務回應可能會有所不同。Step Functions Local 不驗證模擬響應結構是否符合實際的服務響應結構。在測試之前，您必須確保模擬響應符合實際響應。若要檢閱服務回應的結構，您可以使用 Step Functions 執行實際的服務呼叫，或檢視這些服務的說明文件。

第 2 步：提供模擬配置文件 Step Functions 本地

您可以通過以下方式之一提供模擬配置文件 Step Functions 局部：

Docker

> **Note**
>
> 如果您使用的是 Step Functions 數本地的 Docker 版本，則只能使用環境變量提供模擬配置文件。此外，您必須在初始伺服器啟動時，將模擬配置檔案掛載到 Step Functions 本機容器上。

將模擬配置文件安裝到 Step Functions 本地容器中的任何目錄。然後，設置一個名為的環境變量，SFN_MOCK_CONFIG該變量包含容器中的 mock 配置文件的路徑。這種方法使模擬配置文件被命名為任何只要環境變量包含文件路徑和名稱。

下面的命令顯示了啟動 Docker 圖像的格式。

```
docker run -p 8083:8083
--mount type=bind,readonly,source={absolute path to mock config file},destination=/
home/StepFunctionsLocal/MockConfigFile.json
-e SFN_MOCK_CONFIG="/home/StepFunctionsLocal/MockConfigFile.json" amazon/aws-
stepfunctions-local
```

下列範例會使用指令來啟動 Docker 影像。

```
docker run -p 8083:8083
--mount type=bind,readonly,source=/Users/admin/Desktop/workplace/
MockConfigFile.json,destination=/home/StepFunctionsLocal/MockConfigFile.json
```

```
-e SFN_MOCK_CONFIG="/home/StepFunctionsLocal/MockConfigFile.json" amazon/aws-
stepfunctions-local
```

JAR File

使用以下方法之一來提供模擬配置文件 Step Functions 局部：

- 將模擬配置文件放在與相同的目錄中Step FunctionsLocal.jar。使用此方法時，您必須命名模擬配置文件MockConfigFile.json。
- 在運行 Step Functions 數本地的會話中，將名為SFN_MOCK_CONFIG的環境變量設置為模擬配置文件的完整路徑。這種方法使模擬配置文件只要環境變量包含其文件路徑和名稱被命名為任何東西。在下面的例子中，SFN_MOCK_CONFIG變量被設置為指向一個名為的模擬配置文件EnvSpecifiedMockConfig.json，位於目/home/workspace錄中。

```
export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
```

> **Note**
>
> - 如果您不提供環境變量 SFN_MOCK_CONFIG Step Functions 數本地，默認情況下，它將嘗試讀取從中啟動 Step Functions 數本地目錄MockConfigFile.json中命名的模擬配置文件。
> - 如果將模擬配置文件放在相同的目錄中，Step FunctionsLocal.jar並設置環境變量SFN_MOCK_CONFIG，Step Functions Local 將讀取由環境變量指定的文件。

步驟 3：執行模擬服務整合測試

建立並提供模擬組態檔案給 Step Functions Local 之後，請使用模擬服務整合執行在模擬設定檔中設定的狀態機器。然後使用 API 動作檢查執行結果。

1. 根據模擬配置文件中先前提到的定義創建一個狀態機。

```
aws stepfunctions create-state-machine \
    --endpoint http://localhost:8083 \
    --definition "{\"Comment\":\"Thisstatemachineiscalled:LambdaSQSIntegration
\",\"StartAt\":\"LambdaState\",\"States\":{\"LambdaState\":{\"Type\":
\"Task\",\"Resource\":\"arn:aws:states:::lambda:invoke\",\"Parameters
\":{\"Payload.$\":\"$\",\"FunctionName\":\"arn:aws:lambda:us-
```

```
east-1:123456789012:function:HelloWorldFunction\"},\"Retry\":[{\"ErrorEquals
\":[\"States.ALL\"],\"IntervalSeconds\":2,\"MaxAttempts\":3,\"BackoffRate
\":2}],\"Next\":\"SQSState\"},\"SQSState\":{\"Type\":\"Task\",\"Resource\":
\"arn:aws:states:::sqs:sendMessage\",\"Parameters\":{\"QueueUrl\":\"https://
sqs.us-east-1.amazonaws.com/123456789012/myQueue\",\"MessageBody.$\":\"$\"},\"End
\":true}}}" \
    --name "LambdaSQSIntegration" --role-arn "arn:aws:iam::123456789012:role/
service-role/LambdaSQSIntegration"
```

2. 使用模擬服務集成運行狀態機。

   要使用模擬配置文件，請在模擬配置文件中配置的狀態機器上進行 StartExecution API 調用。若要這麼做，請將尾碼附加至使用的狀態機器 ARN。#*test_name* StartExecution *test_name*是一個測試用例，該用例在同一模擬配置文件中為狀態機配置。

   以下命令是使用LambdaSQSIntegration狀態機和模擬配置的示例。在此範例中，使用中定義的HappyPath測試執行LambdaSQSIntegration狀態機器步驟 1：在模擬配置文件中指定模擬服務集成。該HappyPath測試包含用於處理模擬服務集成調用的執行配置，以LambdaState及使用MockedLambdaSuccess和模MockedSQSSuccess擬服務響應進行的SQSState狀態。

   ```
   aws stepfunctions start-execution \
       --endpoint http://localhost:8083 \
       --name executionWithHappyPathMockedServices \
       --state-machine arn:aws:states:us-
   east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
   ```

3. 檢視狀態機器執行回應。

   StartExecution使用模擬服務集成測試調用的響應與StartExecution正常調用的響應相同，該響應返回執行 ARN 和開始日期。

   以下是StartExecution使用模擬服務整合測試呼叫的範例回應：

   ```
   {
      "startDate":"2022-01-28T15:03:16.981000-05:00",
      "executionArn":"arn:aws:states:us-
   east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
   }
   ```

4. 透過進行ListExecutions、DescribeExecution或 GetExecutionHistory API 呼叫來檢查執行結果。

```
aws stepfunctions get-execution-history \
    --endpoint http://localhost:8083 \
    --execution-arn arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
```

下列範例會示範GetExecutionHistory使用步驟 2 所示範例回應的執行 ARN 呼叫的部分回應。在此範例中，LambdaState和的輸出SQSState是在 mock 組態檔案中MockedLambdaSuccess定義的模擬資料。MockedSQSSuccess此外，模擬資料的使用方式與使用實際服務整合呼叫所傳回的資料相同。此外，在這個例子中，從LambdaState的輸出SQSState作為輸入傳遞到。

```
{
    "events": [
        ...
        {
            "timestamp": "2021-12-02T19:39:48.988000+00:00",
            "type": "TaskStateEntered",
            "id": 2,
            "previousEventId": 0,
            "stateEnteredEventDetails": {
                "name": "LambdaState",
                "input": "{}",
                "inputDetails": {
                    "truncated": false
                }
            }
        },
        ...
        {
            "timestamp": "2021-11-25T23:39:10.587000+00:00",
            "type": "LambdaFunctionSucceeded",
            "id": 5,
            "previousEventId": 4,
            "lambdaFunctionSucceededEventDetails": {
                "output": "{\"statusCode\":200,\"body\":\"\\\"Hello from Lambda!\\
\"\"}",
                "outputDetails": {
                    "truncated": false
                }
            }
        },
```

```
        ...
            "timestamp": "2021-12-02T19:39:49.464000+00:00",
            "type": "TaskStateEntered",
            "id": 7,
            "previousEventId": 6,
            "stateEnteredEventDetails": {
                "name": "SQSState",
                "input": "{\"statusCode\":200,\"body\":\"\\\"Hello from Lambda!\\
\"\"}",
                "inputDetails": {
                    "truncated": false
                }
            }
        },
        ...
        {
            "timestamp": "2021-11-25T23:39:10.652000+00:00",
            "type": "TaskSucceeded",
            "id": 10,
            "previousEventId": 9,
            "taskSucceededEventDetails": {
                "resourceType": "sqs",
                "resource": "sendMessage",
                "output": "{\"MD5OfMessageBody\":\"3bcb6e8e-7h85-4375-
b0bc-1a59812c6e51\",\"MessageId\":\"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51\"}",
                "outputDetails": {
                    "truncated": false
                }
            }
        },
        ...
    ]
}
```

模擬服務整合的組態檔

要使用模擬服務集成，您必須首先創建一個名為MockConfigFile.json包含模擬配置的模擬配置文件。然後提供步驟函數本地與模擬配置文件。此配置文件定義了測試用例，其中包含使用模擬服務集成響應的模擬狀態。以下部分包含模擬配置結構的信息，其中包括模擬狀態和模擬響應：

主題

- 引入模擬配置的結構

- 設定模擬服務整合

引入模擬配置的結構

模擬配置是包含以下頂級字段的 JSON 對象：

- StateMachines-此物件的欄位代表設定為使用模擬服務整合的狀態機器。
- MockedResponse-此物件的欄位代表服務整合呼叫的模擬回應。

下面是一個模擬配置文件的一個例子，其中包括一個StateMachine定義和MockedResponse。

```
{
  "StateMachines":{
    "LambdaSQSIntegration":{
      "TestCases":{
        "HappyPath":{
          "LambdaState":"MockedLambdaSuccess",
          "SQSState":"MockedSQSSuccess"
        },
        "RetryPath":{
          "LambdaState":"MockedLambdaRetry",
          "SQSState":"MockedSQSSuccess"
        },
        "HybridPath":{
          "LambdaState":"MockedLambdaSuccess"
        }
      }
    }
  },
  "MockedResponses":{
    "MockedLambdaSuccess":{
      "0":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
          }
        }
      }
    },
    "LambdaMockedResourceNotReady":{
```

```
      "0":{
        "Throw":{
          "Error":"Lambda.ResourceNotReadyException",
          "Cause":"Lambda resource is not ready."
        }
      }
    },
    "MockedSQSSuccess":{
      "0":{
        "Return":{
          "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
          "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
        }
      }
    },
    "MockedLambdaRetry":{
      "0":{
        "Throw":{
          "Error":"Lambda.ResourceNotReadyException",
          "Cause":"Lambda resource is not ready."
        }
      },
      "1-2":{
        "Throw":{
          "Error":"Lambda.TimeoutException",
          "Cause":"Lambda timed out."
        }
      },
      "3":{
        "Return":{
          "StatusCode":200,
          "Payload":{
            "StatusCode":200,
            "body":"Hello from Lambda!"
          }
        }
      }
    }
  }
}
```

模擬配置字段參考

下列各節說明您必須在 mock 組態中定義的頂層物件欄位。

- StateMachines
- MockedResponses

StateMachines

`StateMachines`物件會定義哪些狀態機器將使用模擬服務整合。每個狀態機的組態都會以的最上層欄位表示`StateMachines`。字段名稱是狀態機的名稱，value 是包含一個名為的單個字段的對象`TestCases`，其字段代表該狀態機的測試用例。

以下語法顯示了具有兩個測試用例的狀態機：

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

的欄位代`TestCases`表狀態機器的個別測試案例。每個狀態機器的每個測試用例的名稱必須是唯一的，並且每個測試用例的值是一個對象，指定用於狀態機器中任務狀態的模擬響應。

以下實例將兩個狀態`TestCase`鏈接到兩個`Task`狀態`MockedResponses`：

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
  "AnotherTaskState": "AnotherMockedResponse"
}
```

MockedResponses

`MockedResponses`是一個包含多個具有唯一字段名稱的模擬響應對象的對象。模擬響應對象定義了模擬任務狀態的每次調用成功的結果或錯誤輸出。您可以使用個別的整數字串 (例如「0」、「1」、「2」和「3」) 或包含整數 (例如「0-1」、「2-3」) 來指定叫用編號。

當你模擬一個 Task 時，你必須為每次調用指定一個模擬響應。回應必須包含名為`Return`或`Throw`其值為模擬 Task 叫用的結果或錯誤輸出的單一欄位。如果未指定模擬回應，狀態機器執行將會失敗。

以下是MockedResponse與Throw和Return物件的範例。在此範例中，執行狀態機器的前三次、傳回中"0-2"指定的回應，而第四次執行狀態機時，會傳回中指定"3"的回應。

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> 如果您正在使用Map狀態，並且想要確保Map狀態的可預測回應，請將值設定maxConcurrency為 1。如果您設定大於 1 的值，Step Functions Local 會同時執行多個反覆項目，這會導致反覆運算間狀態的整體執行順序無法預測。這可能會進一步導致 Step Functions Local 對從一個執行到下一個執行的迭代狀態使用不同的模擬響應。

傳回

Return表示為MockedResponse物件的欄位。它指定了一個模擬任務狀態的成功結果。

以下是Return物件的範例，其中包含用於呼叫 Lambda 函數Invoke的模擬回應：

```
"Return": {
  "StatusCode": 200,
  "Payload": {
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
}
```

投擲

Throw表示為MockedResponse物件的欄位。它指定一個失敗的任務的錯誤輸出。的值Throw必須是包含Error和具有字串值之Cause欄位的物件。此外，您在Error欄位中指定的字串值MockConfigFile.json必須與狀態機器Retry和Catch區段中處理的錯誤相符。

以下是Throw物件的範例，其中包含用於呼叫 Lambda 函數Invoke的模擬回應：

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
}
```

設定模擬服務整合

您可以模擬使用步驟功能本地任何服務集成。但是，步驟函數本地不會強制模擬與真實 API 相同。模擬任務永遠不會調用服務端點。如果您未指定模擬回應，Task 將嘗試呼叫服務端點。此外，當您使用模擬任務時，步驟函數本地將自動生成任務令牌.waitForTaskToken。

Step Functions 的最佳做法

以下實作 AWS Step Functions 工作流程的最佳實務，可協助您優化實作的效能。

主題

使用超時來避免卡住的執行

根據預設，Amazon 州語言不會指定狀態機器定義的逾時。如果沒有明確的逾時，Step Functions 通常僅依賴活動工作者的回應來知道任務已完成。如果出現錯誤，並且未為`Activity`或`Task`狀態指定`TimeoutSeconds`欄位，則執行會卡住等待永遠不會出現的回應。

為了避免這種情況，請在狀態機器中創建時指定合理`Task`的超時時間。例如：

```
"ActivityState": {
  "Type": "Task",
  "Resource": "arn:aws:states:us-east-1:123456789012:activity:HelloWorld",
  "TimeoutSeconds": 300,
  "Next": "NextState"
}
```

如果您使用帶有任務令牌的回調（.waitForTaskToken），我們建議您使用活動訊號，並在`Task`狀態定義中新增`HeartbeatSeconds`欄位。您可以設定`HeartbeatSeconds`為小於工作逾時，所以如果您的工作流程因活動訊號錯誤而失敗，您就會知道這是因為工作失敗，而不是工作需要很長時間才能完成。

```
{
  "StartAt": "Push to SQS",
```

```
  "States": {
    "Push to SQS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
      "HeartbeatSeconds": 600,
      "Parameters": {
        "MessageBody": { "myTaskToken.$": "$$.Task.Token" },
        "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/push-based-queue"
      },
      "ResultPath": "$.SQS",
      "End": true
    }
  }
}
```

如需詳細資訊，請參閱 Amazon 州語言文件任務中的。

> **Note**
>
> 您可以使用 Amazon States 語言定義中的`TimeoutSeconds`欄位，為狀態機器設定逾時。如需詳細資訊，請參閱国家机结构。

使用 Amazon S3 ARN 而不是傳遞大型有效載荷

在狀態之間傳遞大型資料承載的執行作業可能會被終止。如果您在各個狀態之間傳遞的資料可能會增長到超過 256 KB，請使用 Simple Storage Service (Amazon S3) 存放資料，然後剖析`Payload`參數中儲存貯體的 Amazon 資源名稱 (ARN) 以取得儲存貯體名稱和金鑰值。或者，調整您的實作，以便在您的執行作業中傳遞較小的承載。

在下列範例中，狀態機器會將輸入傳遞至AWS Lambda函數，該函數會處理 Amazon S3 儲存貯體中的 JSON 檔案。執行此狀態機器之後，Lambda 函數會讀取 JSON 檔案的內容，並傳回檔案內容做為輸出。

建立 Lambda 函數

下列名為的 Lambda 函數*`pass-large-payload`*會讀取儲存在特定 Amazon S3 儲存貯體中的 JSON 檔案的內容。

> **Note**
>
> 建立此 Lambda 函數之後，請務必提供其 IAM 角色的適當權限，以便從 Amazon S3 儲存貯體讀取。例如，將亞馬遜 S3 ReadOnlyAccess 權限附加到 Lambda 函數的角色。

```
import json
import boto3
import io
import os

s3 = boto3.client('s3')

def lambda_handler(event, context):
    event = event['Input']
    final_json = str()

    s3 = boto3.resource('s3')
    bucket = event['bucket'].split(':')[-1]
    filename = event['key']
    directory = "/tmp/{}".format(filename)

    s3.Bucket(bucket).download_file(filename, directory)

    with open(directory, "r") as jsonfile:

        final_json = json.load(jsonfile)

    os.popen("rm -rf /tmp")

    return final_json
```

建立狀態機

下列狀態機會叫用您先前建立的 Lambda 函數。

```
{
   "StartAt":"Invoke Lambda function",
   "States":{
      "Invoke Lambda function":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
```

```
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-2:123456789012:function:pass-large-
payload",
            "Payload":{
               "Input.$":"$"
            }
         },
         "OutputPath": "$.Payload",
         "End":true
      }
   }
}
```

您可以將資料儲存在 Amazon S3 儲存貯體中，然後在Payload參數中傳遞儲存貯體的 Amazon 資源名稱 (ARN) 以取得儲存貯體名稱和金鑰值，而不是在輸入中傳遞大量資料。然後，您的 Lambda 函數就可以使用該 ARN 直接存取資料。以下是狀態機器執行的範例輸入，其中資料存放data.json在名為的 Amazon S3 儲存貯體中*large-payload-json*。

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

避免達到歷史記錄配額

AWS Step Functions在執行事件歷史記錄中有 25,000 個項目的硬配額。當執行達到 24,999 個事件時，它會等待下一個事件發生。

- 如果事件編號 25,000 為ExecutionSucceeded，則執行成功完成。
- 如果事件編號 25,000 不是ExecutionSucceeded，則會記錄ExecutionFailed事件，並且狀態機器執行失敗，因為達到歷史記錄限制

若要避免長時間執行的執行達到此配額，您可以嘗試下列其中一種因應措施：

- 在分散式模式中使用「對映」狀態。在此模式下，Map狀態會將每個反覆作為子工作流程執行執行執行執行來執行，最多可達 10,000 個 parallel 子工作流程執行的高並行性。每個子工作流程執行都有自己的獨立執行記錄與父工作流程的執行記錄。
- 直接從執行中的狀態開始新的Task狀態機執行。若要啟動此類巢狀工作流程執行，請在父狀態機器中使用 Step Functions 的 StartExecution API 動作以及必要的參數。如需有關使用巢狀工作

流程的詳細資訊，請參閱從任務狀態開始工作流程執行或使用 Step Functions API 動作繼續新的執行教學課程。

> **Tip**
>
> 若要將巢狀工作流程的範例部署到您的AWS 帳戶，請參閱單元 13-巢狀 Express 工作流程。

- 實施一種模式，該模式使用可以啟動狀態機器的新執行的AWS Lambda功能，以跨多個工作流程執行分割正在進行的工作。如需詳細資訊，請參閱使用 Lambda 函數繼續新的執行教學課程。

處理 Lambda 務例外

AWS Lambda 可以偶爾體驗暫時性服務錯誤。在此情況下，呼叫 Lambda 會導致 500 個錯誤，例如`ClientExecutionTimeoutException`、`ServiceException`、`AWSLambdaException`、或`SdkClientException`。最佳做法是在狀態機器中主動處理這些例外狀況，以`Retry`叫用 Lambda 函數或錯誤。`Catch`

Lambda 錯誤報告為`Lambda.`*`ErrorName`*。若要重試 Lambda 服務例外狀況錯誤，您可以使用下列`Retry`程式碼。

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Lambda 中未處理的錯誤會報告為錯誤輸出`Lambda.Unknown`中。這些包括 out-of-memory 錯誤和功能超時。您可以在`Lambda.UnknownStates.ALL`、或上進行比對`States.TaskFailed`來處理這些錯誤。當 Lambda 達到調用的最大數量時，錯誤為。`Lambda.TooManyRequestsException`如需 Lambda 函數錯誤的詳細資訊，請參閱AWS Lambda開發人員指南中的錯誤處理和自動重試。

如需詳細資訊，請參閱下列內容：

- 發生錯誤後重試
- 使用 Step Functions 狀態機處理錯誤條件
- Lambda 調用錯誤

輪詢活動任務時避免延遲

該 GetActivityTask API 旨在提供taskToken一次。如果 taskToken 在與活動工作者通訊時遺失，可以封鎖數個 GetActivityTask 請求 60 秒來等待回應，直到 GetActivityTask 逾時為止。

如果您只有少量輪詢在等待回應，則所有請求可能都會排在封鎖的請求後面並且停止。但是，如果您對每個活動 Amazon 資源名稱（ARN）都有大量未完成的民意調查，並且某些比例的請求卡在等待中，則還有更多可以獲得taskToken並開始處理工作。

對於生產系統，我們建議每個活動 ARN 隨時至少都有 100 個開啟的輪詢。如果有一個輪詢遭到封鎖，而且有一部分的輪詢排在其後面，則有更多請求仍會收到 GetActivityTask 以在 taskToken 請求遭到封鎖時處理工作。

若要在輪詢任務時避免這幾種延遲問題：

- 在您的活動工作者實作中，以個別的執行緒實作您的輪詢器。
- 每個活動 ARN 隨時至少都有 100 個開啟的輪詢。

> **Note**
>
> 每個 ARN 擴展至 100 個開放輪詢可能所費不貲。例如，每個 ARN 進行 100 個 Lambda 函數輪詢的成本比具有 100 個輪詢執行緒的單一 Lambda 函數高 100 倍。若要降低延遲並將成本降至最低，請使用具有非同步 I/O 的語言，然後針對每個工作人員實作多個輪詢執行緒。如需輪詢器執行緒與工作執行緒不同的範例活動工作者，請參閱 Ruby 中的範例活動工作者。

如需活動和活動工作者的詳細資訊，請參閱 活動。

選擇標準或快速工作流程

AWS Step Functions 提供標準工作流程做為預設工作流程類型，此外還提供快速工作流程的選項。

當您需要長時間執行、耐用且可稽核的工作流程時，您可以選擇標準工作流程，或選擇 Express 工作流程來處理大量事件處理工作負載。根據您選取的 Type，狀態機器執行的行為會有所不同。建立狀態機器後，您選擇的 Type 將無法變更。

- 如需標準工作流程和快速工作流程之間差異的詳細資訊，請參閱標準與快速工作流程。
- 如需使用 Step Functions 建置無伺服器工作流程時最佳化成本的資訊，請參閱使用快速工作流程最佳化。

亞馬遜 CloudWatch 日誌資源政策大小限制

CloudWatch 記錄檔資源策略的長度限制為 5120 個字元。當 CloudWatch 記錄檔偵測到原則接近此大小限制時，會自動啟用以開頭的記錄群組/aws/vendedlogs/。

當您建立啟用記錄的狀態機器時，Step Functions 必須使用您指定的記 CloudWatch 錄群組來更新記錄資源原則。若要避免達到 CloudWatch 記錄檔資源原則大小限制，請在 CloudWatch 記錄檔群組名稱前面加上/aws/vendedlogs/。當您在 Step Functions 主控台中建立記錄群組時，記錄群組名稱的前置詞會加上。/aws/vendedlogs/states如需詳細資訊，請參閱啟用來自某些AWS服務的記錄。

如果您無法存取 CloudWatch 記錄檔，請參閱以取得相Unable to access the CloudWatch Logs關資訊。

AWS Step Functions 搭配其他服務使用

了解如何使 AWS Step Functions用協調其他 AWS 服務 或呼叫第三方 API。

主題

致電其他 AWS 服務

AWS Step Functions 與 AWS 服務整合，可讓您從工作流程中呼叫每個服務的 API 動作。您可以使用 Step Functions 的 AWS SDK 整合，直接從狀態機調用 200 多個 AWS 服務中的任何一個，從而使您可以訪問超過九千個 API 操作。或者，您也可以使用 Step Functions 的最佳化整合，每個整合都經過自訂，為您的工作流程提供特殊功能。某些 API 動作可在這兩種整合類型中使用。在此情況下，建議您使用「最佳化」整合。

您可以直接從 Amazon 州語言中的某個`Task`州協調這些服務。例如，使用 Step Functions，您可以呼叫其他服務：

- 調用一個 AWS Lambda 函數。
- 執行 AWS Batch 工作，然後根據結果執行不同的動作。
- 插入或從亞 Amazon DynamoDB 項目。
- 運行 Amazon Elastic Container Service (Amazon ECS) 任務，並等待它完成。
- 發佈到 Amazon Simple Notification Service (Amazon SNS) 中的主題。
- 在 Amazon Simple Queue Service (Amazon SQS) 中發送消息。
- 管理 AWS Glue 或 Amazon 的任務 SageMaker。
- 建立用於執行 Amazon EMR 任務的工作流程。
- 啟動 AWS Step Functions 工作流程執行。

最佳化整合

Step Functions 已自訂最佳化整合，以提供工作流程前後關聯的特殊功能。例如，Lambda 會Invoke將其 API 輸出從逸出的 JSON 轉換為 JSON 物件。 AWS BatchSubmitJob可讓您暫停執行，直到工作完成為止。第一組優化集成在 2018 年發布，現在有五十多個 API。

AWS SDK 整合功能

AWS SDK 整合的運作方式與使用 AWS SDK 的標準 API 呼叫完全相同。它們提供了直接從您的狀態機器定義中調用超過二百個 AWS 服務中的九千多個 API 的能力。 AWS SDK 整合功能於 2021 年推出。

整合模式支援

標準工作流程和 Express 工作流程支援相同的整合，但不支援相同的整合模式。

- 每個集成的優化集成模式支持都不同。
- Express Job 流程不支援執行工作 (.sync) 或等待回呼 (. waitForTask令牌）。
- 如需詳細資訊，請參閱 標準與快速工作流程。

Standard Workflows

支援的服務整合

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
最佳化整合	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS SDK 整合功能	超過二百個	✓		✓

Express Workflows

支援的服務整合

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
最佳化整合	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS SDK 整合功能	超過二百個	✓		

跨帳戶存取權

Step Functions 提供跨帳戶存取工作流程中不同設定 AWS 帳戶 的資源。使用 Step Functions 服務整合，即使 AWS 服務 不支援以 AWS 資源為基礎的政策或跨帳戶呼叫，您也可以叫用任何跨帳戶資源。

如需詳細資訊，請參閱 存取工作流程 AWS 帳戶 中其他資源。

AWS SDK 服務整合

AWS Step Functions 與整合 AWS 服務，可讓您從工作流程中呼叫每個服務的 API 動作。您可以使用 Step 函數的 AWS SDK 集成從狀態機調用幾乎所有 AWS 服務的 API 操作。您也可以使用 Step Functions 的最佳化整合，每個整合都經過自訂，為您的工作流程提供特殊功能。

某些服務或 SDK 可能無法作為 AWS SDK 整合提供，無論是暫時還是永久。最近發行的服務在稍後更新之前可能沒有可用的 SDK 互動。某些服務需要自訂組態，例如指定客戶特定的端點，這可能更適合最佳化的整合。其他 SDK 不適合在工作流程中使用，例如用於流式傳輸音頻或視頻的 SDK。最後，某些服務可能會被扣留，直到它們通過 Step Functions 執行的某些內部驗證。

> **Tip**
>
> 若要將使用 AWS SDK 整合的工作流程範例部署至您的工作流程 AWS 帳戶，請參閱 AWS Step Functions 工作坊中的模組 9- AWS SDK 服務整合。

主題

使用 AWS SDK 服務整合

若要使用 AWS SDK 整合，您可以指定服務名稱和 API 呼叫，以及選擇性地指定服務整合模式。

> **Note**
>
> - 中的參數Step Functions會在中表示 PascalCase，即使原生服務 API 位於駝峰中也是如此。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。
> - 對於接受列舉參數的 API 動作 (例如 Amazon EC2 的 `DescribeLaunchTemplateVersions` API 動作)，請指定參數名稱的複數版本。例如，`Filters`為 `DescribeLaunchTemplateVersions` API 動作的`Filter.N`參數指定。

您可以直接從任務狀態`Resource`欄位中的 Amazon 州語言呼叫 AWS SDK 服務。若要這麼做，請使用下列語法：

`arn:aws:states:::aws-sdk:`*`serviceName`*`:`*`apiAction`*`.`*`[serviceIntegrationPattern]`*

例如，對於 Amazon EC2，您可以使用`arn:aws:states:::aws-sdk:ec2:describeInstances`. 這將返回為 Amazon EC2 說明實例 API 調用定義的輸出。

如果 AWS SDK 整合遇到錯誤，則產生的「錯誤」欄位將由服務名稱和錯誤名稱組成，並以句號字元分隔：*`ServiceName`*`.`*`ErrorName`*。無論是服務名稱和錯誤名稱是在帕斯卡的情況下。您也可以在「工作」狀態的「資源」欄位中以小寫形式查看服務名稱。您可以在目標服務的 API 參考文件中找到潛在的錯誤名稱。

例如，您可以使用 `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK 整合。例如`ResourceNotFoundException`，AWS Private Certificate Authority API 參考指出 `DeleteCertificateAuthority` API 動作可能會產生一個。要處理此錯誤，因此您可以`AcmPca.ResourceNotFoundException`在任務狀態的「檢索器」或「捕獲器」中指定「錯誤」。如需「Step Functions」中錯誤處理的詳細資訊，請參閱Step Functions 中的錯誤處理。

Step Functions 無法自動產生 AWS SDK 整合的 IAM 政策。建立狀態機後，您將需要導覽至 IAM 主控台並設定角色政策。如需詳細資訊，請參閱整合式服務的 IAM 政策。

如需如何使用 AWS SDK 整合的範例，請參閱使用 AWS SDK 服務整合收集 Amazon S3 儲存貯體資訊教學課程。

支援的 AWS SDK 服務整合

下表列出 Step Functions 支援的 AWS SDK 服務整合。狀*Task*態資源欄會列出使用左側 [服務名稱] 欄中指定的服務整合時，呼叫特定 API 動作的語法。「支援日期」欄提供支援服務整合的日期相關資訊。此外，右側的「例外前置字元」欄會列出每個服務整合的例外前置字元。當您錯誤地執行與 Step Functions 的 AWS SDK 服務整合時，會產生這些例外狀況前置詞存在於例外狀況中。

> **Note**
>
> - 標有 *** 的服務具有此時 Step Functions 不支援的 API 動作。如需服務不支援之動作的相關資訊，請參閱支援服務不支援的 API 動作表格。
> - 如需有關每次啟動以擴展 AWS SDK 整合支援的更新資訊，請參閱變更支援 AWS SDK 整合的記錄。

> **Important**
>
> API 操作支持按季度發布。已支援的動作 (例如新參數) 的更新可能無法立即使用。

服務名稱	**Task**狀態資源	支援的日期	異常前綴
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric:` *[apiAction]*	2024年1月18日	AppFabric
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi:` *[apiAction]*	2024年1月18日	B2Bi

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS 資料匯出	arn:aws:states:::aws-sdk:bcmdataexports: *[apiAction]*	2024年1月18日	BcmDataExports
Amazon Bedrock	arn:aws:states:::aws-sdk:bedrock: *[apiAction]*	2024年1月18日	Bedrock
Amazon Bedrock Agents	arn:aws:states:::aws-sdk:bedrockagent: *[apiAction]*	2024年1月18日	BedrockAgent
Amazon Bedrock Runtime Agents	arn:aws:states:::aws-sdk:bedrockagentruntime: *[apiAction]*	2024年1月18日	BedrockAgentRuntime
Amazon Bedrock Runtime	arn:aws:states:::aws-sdk:bedrockruntime: *[apiAction]*	2024年1月18日	BedrockRuntime
AWS Clean Rooms	arn:aws:states:::aws-sdk:cleanroomsml: *[apiAction]*	2024年1月18日	CleanRoomsMl
Amazon CloudFront KeyValueStore	arn:aws:states:::aws-sdk:cloudfrontkeyvaluestore: *[apiAction]*	2024年1月18日	CloudFrontKeyValueStore
CodeGuru Security	arn:aws:states:::aws-sdk:codegurusecurity: *[apiAction]*	2024年1月18日	CodeGuruSecurity

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS 成本最佳化中心	arn:aws:states:::aws-sdk:costoptimizationhub: *[apiAction]*	2024年1月18日	CostOptimizationHub
Amazon DataZone	arn:aws:states:::aws-sdk:datazone: *[apiAction]*	2024年1月18日	DataZone
Amazon EKS Auth	arn:aws:states:::aws-sdk:eksauth: *[apiAction]*	2024年1月18日	EksAuth
AWS Entity Resolution	arn:aws:states:::aws-sdk:entityresolution: *[apiAction]*	2024年1月18日	EntityResolution
AWS 免費方案	arn:aws:states:::aws-sdk:freetier: *[apiAction]*	2024年1月18日	FreeTier
Amazon Inspector Scan	arn:aws:states:::aws-sdk:inspectorscan: *[apiAction]*	2024年1月18日	InspectorScan
AWS Launch Wizard	arn:aws:states:::aws-sdk:launchwizard: *[apiAction]*	2024年1月18日	LaunchWizard
Amazon Managed Blockchain Query	arn:aws:states:::aws-sdk:managedblockchainquery: *[apiAction]*	2024年1月18日	ManagedBlockchainQuery

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Elemental MediaPackage V2	arn:aws:states:::aws-sdk:mediapackagev2: *[apiAction]*	2024年1月18日	MediaPackageV2
AWS HealthImaging	arn:aws:states:::aws-sdk:medicalimaging: *[apiAction]*	2024年1月18日	MedicalImaging
Network Manager	arn:aws:states:::aws-sdk:networkmanager: *[apiAction]*	2024年1月18日	NetworkManager
AWS Payment Cryptography	arn:aws:states:::aws-sdk:paymentcryptography: *[apiAction]*	2024年1月18日	PaymentCryptography
AWS Payment Cryptography Data	arn:aws:states:::aws-sdk:paymentcryptographydata: *[apiAction]*	2024年1月18日	PaymentCryptographyData
AWS Private CA Connector for Active Directory	arn:aws:states:::aws-sdk:pcaconnectorad: *[apiAction]*	2024年1月18日	PcaConnectorAd
Amazon Q Business	arn:aws:states:::aws-sdk:qbusiness: *[apiAction]*	2024年1月18日	QBusiness
Amazon Q Connect	arn:aws:states:::aws-sdk:qconnect: *[apiAction]*	2024年1月18日	QConnect

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS re:Post	arn:aws:states:::aws-sdk:repostspace: *[apiAction]*	2024年1月18日	Repostspace
Amazon Timestream Query	arn:aws:states:::aws-sdk:timestreamquery: *[apiAction]*	2024年1月18日	TimestreamQuery
Amazon Timestream Write	arn:aws:states:::aws-sdk:timestreamwrite: *[apiAction]*	2024年1月18日	TimestreamWrite
Trusted Advisor	arn:aws:states:::aws-sdk:trustedadvisor: *[apiAction]*	2024年1月18日	TrustedAdvisor
Verified Permissions	arn:aws:states:::aws-sdk:verifiedpermissions: *[apiAction]*	2024年1月18日	VerifiedPermissions
Amazon WorkSpaces Thin Client	arn:aws:states:::aws-sdk:workspacesthinclient: *[apiAction]*	2024年1月18日	WorkSpacesThinClient
AWS CloudTrail Data	arn:aws:states:::aws-sdk:cloudtrailddata: *[apiAction]*	2023 年 6 月 16 日	CloudTrailData
Amazon CloudWatch Internet Monitor	arn:aws:states:::aws-sdk:internetmonitor: *[apiAction]*	2023 年 6 月 16 日	InternetMonitor

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Interactive Video Service RealTime	arn:aws:states:::aws-sdk:ivsrealtime: *[apiAction]*	2023 年 6 月 16 日	IvsRealTime
AWS IoT TwinMaker	arn:aws:states:::aws-sdk:iottwinmaker: *[apiAction]*	2023 年 6 月 16 日	IoTTwinMaker
Amazon OpenSearch Ingestion	arn:aws:states:::aws-sdk:osis: *[apiAction]*	2023 年 6 月 16 日	Osis
AWS Telco Network Builder	arn:aws:states:::aws-sdk: **tnb**:*[apiAction]*	2023 年 6 月 16 日	Tnb
Amazon VPC Lattice	arn:aws:states:::aws-sdk:vpclattice: *[apiAction]*	2023 年 6 月 16 日	VpcLattice
Amazon Chime Media Pipelines	arn:aws:states:::aws-sdk:chimesdkmediapipelines: *[apiAction]*	2023 年 2 月 17 日	ChimeSdkMediaPipelines
Amazon Chime Voice	arn:aws:states:::aws-sdk:chimesdkvoice: *[apiAction]*	2023 年 2 月 17 日	ChimeSdkVoice
Amazon CodeCatalyst	arn:aws:states:::aws-sdk:codecatalyst: *[apiAction]*	2023 年 2 月 17 日	CodeCatalyst
Amazon Connect Cases	arn:aws:states:::aws-sdk:connectcases: *[apiAction]*	2023 年 2 月 17 日	ConnectCases

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon DocumentDB Elastic Clusters	arn:aws:states:::aws-sdk:docdbelastic: *[apiAction]*	2023 年 2 月 17 日	DocDbElastic
Amazon EMR Serverless	arn:aws:states:::aws-sdk:emrserverless: *[apiAction]*	2023 年 2 月 17 日	EmrServerless
Amazon IVS Chat	arn:aws:states:::aws-sdk:ivs: *[apiAction]*	2023 年 2 月 17 日	Ivs
Amazon Kendra Intelligent Ranking	arn:aws:states:::aws-sdk:kendraranking: *[apiAction]*	2023 年 2 月 17 日	KendraRanking
AWS HealthOmics	arn:aws:states:::aws-sdk:omics: *[apiAction]*	2023 年 2 月 17 日	Omics
Amazon Redshift Serverless	arn:aws:states:::aws-sdk:redshiftserverless: *[apiAction]*	2023 年 2 月 17 日	RedshiftServerless
Amazon Security Lake	arn:aws:states:::aws-sdk:securitylake: *[apiAction]*	2023 年 2 月 17 日	SecurityLake
AWS Backup Storage	arn:aws:states:::aws-sdk:backupstorage: *[apiAction]*	2023 年 2 月 17 日	BackupStorage
AWS Clean Rooms	arn:aws:states:::aws-sdk:cleanrooms: *[apiAction]*	2023 年 2 月 17 日	CleanRooms

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Control Tower	arn:aws:states:::aws-sdk:controltower: *[apiAction]*	2023 年 2 月 17 日	ControlTower
AWS Health	arn:aws:states:::aws-sdk:health: *[apiAction]*	2023 年 2 月 17 日	Health
AWS IoT FleetWise	arn:aws:states:::aws-sdk:iotfleetwise: *[apiAction]*	2023 年 2 月 17 日	IotFleetWise
AWS IoT RoboRunner	arn:aws:states:::aws-sdk:iotrobotrunner: *[apiAction]*	2023 年 2 月 17 日	IotRoboRunner
AWS Mainframe Modernization	arn:aws:states:::aws-sdk:m2: *[apiAction]*	2023 年 2 月 17 日	M2
AWS Migration Hub Orchestrator	arn:aws:states:::aws-sdk:migrationhuborchestrator: *[apiAction]*	2023 年 2 月 17 日	Migration HubOrchestrator
AWS Private 5G	arn:aws:states:::aws-sdk:privatenetworks: *[apiAction]*	2023 年 2 月 17 日	PrivateNetworks
AWS 資源總管	arn:aws:states:::aws-sdk:resourceexplorer2: *[apiAction]*	2023 年 2 月 17 日	ResourceExplorer2
AWS SimSpace Weaver	arn:aws:states:::aws-sdk:simspaceweaver: *[apiAction]*	2023 年 2 月 17 日	SimSpaceWeaver

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Support App	arn:aws:states:::aws-sdk:supportapp: *[apiAction]*	2023 年 2 月 17 日	SupportApp
CloudWatch Observability Access Manager	arn:aws:states:::aws-sdk:oam: *[apiAction]*	2023 年 2 月 17 日	Oam
EventBridge Pipes	arn:aws:states:::aws-sdk:pipes: *[apiAction]*	2023 年 2 月 17 日	Pipes
EventBridge Scheduler	arn:aws:states:::aws-sdk:scheduler: *[apiAction]*	2023 年 2 月 17 日	Scheduler
IAM Roles Anywhere	arn:aws:states:::aws-sdk:rolesanywhere: *[apiAction]*	2023 年 2 月 17 日	RolesAnywhere
Kinesis Video WebRTC Storage	arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage: *[apiAction]*	2023 年 2 月 17 日	KinesisVideoWebRtcStorage
License Manager Linux Subscriptions	arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions: *[apiAction]*	2023 年 2 月 17 日	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	arn:aws:states:::aws-sdk:licensemanagerusersubscriptions: *[apiAction]*	2023 年 2 月 17 日	LicenseManagerUserSubscriptions

服務名稱	Task狀態資源	支援的日期	異常前綴
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless:` *[apiAction]*	2023 年 2 月 17 日	OpenSearchServerless
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift:` *[apiAction]*	2023 年 2 月 17 日	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial:` *[apiAction]*	2023 年 2 月 17 日	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics:` *[apiAction]*	2023 年 2 月 17 日	SageMakerMetrics
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap:` *[apiAction]*	2023 年 2 月 17 日	SsmSap
AWS Account Management	`arn:aws:states:::aws-sdk:account:` *[apiAction]*	2022 年 4 月 19 日	Account
AWS Amplify	`arn:aws:states:::aws-sdk:amplify:` *[apiAction]*	2021 年 9 月 30 日	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh:` *[apiAction]*	2021 年 9 月 30 日	AppMesh

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner:` *[apiAction]*	2021 年 9 月 30 日	AppRunner
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig:` *[apiAction]*	2021 年 9 月 30 日	AppConfig
AWS AppConfig Data	`arn:aws:states:::aws-sdk:appconfigdata:` *[apiAction]*	2022 年 4 月 19 日	AppConfigData
AWS AppSync	`arn:aws:states:::aws-sdk:appsync:` *[apiAction]*	2021 年 9 月 30 日	AppSync
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery:` *[apiAction]* ***	2021 年 9 月 30 日	ApplicationDiscovery
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn:` *[apiAction]*	2021 年 9 月 30 日	Mgn
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager:` *[apiAction]*	2021 年 9 月 30 日	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans:` *[apiAction]*	2021 年 9 月 30 日	AutoScalingPlans
AWS Backup	`arn:aws:states:::aws-sdk:backup:` *[apiAction]*	2021 年 9 月 30 日	Backup

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Backup gateway	arn:aws:states:::aws-sdk:backupgateway: *[apiAction]*	2022 年 4 月 19 日	BackupGateway
AWS Batch	arn:aws:states:::aws-sdk:batch: *[apiAction]*	2021 年 9 月 30 日	Batch
AWS Billing Conductor	arn:aws:states:::aws-sdk:billingconductor: *[apiAction]*	2022 年 7 月 26 日	Billingconductor
AWS Budgets	arn:aws:states:::aws-sdk:budgets: *[apiAction]*	2021 年 9 月 30 日	Budgets
AWS Certificate Manager	arn:aws:states:::aws-sdk:acm: *[apiAction]*	2021 年 9 月 30 日	Acm
AWS Private Certificate Authority	arn:aws:states:::aws-sdk:acmpca: *[apiAction]*	2021 年 9 月 30 日	AcmPca
AWS Cloud Map	arn:aws:states:::aws-sdk:servicediscovery: *[apiAction]*	2021 年 9 月 30 日	ServiceDiscovery
AWS Cloud9	arn:aws:states:::aws-sdk:cloud9: *[apiAction]*	2021 年 9 月 30 日	Cloud9
AWS CloudFormation	arn:aws:states:::aws-sdk:cloudformation: *[apiAction]*	2021 年 9 月 30 日	CloudFormation

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS CloudHSM	arn:aws:states:::aws-sdk:cloudhsm: *[apiAction]*	2021 年 9 月 30 日	CloudHsm
AWS CloudHSM	arn:aws:states:::aws-sdk:cloudhsmv2: *[apiAction]*	2021 年 9 月 30 日	CloudHsmV2
AWS CloudTrail	arn:aws:states:::aws-sdk:cloudtrail: *[apiAction]*	2021 年 9 月 30 日	CloudTrail
AWS Cloud Control	arn:aws:states:::aws-sdk:cloudcontrol: *[apiAction]*	2022 年 4 月 19 日	CloudControl
AWS CodeBuild	arn:aws:states:::aws-sdk:codebuild: *[apiAction]*	2021 年 9 月 30 日	CodeBuild
AWS CodeCommit	arn:aws:states:::aws-sdk:codecommit: *[apiAction]*	2021 年 9 月 30 日	CodeCommit
AWS CodeDeploy	arn:aws:states:::aws-sdk:codedeploy: *[apiAction]* *** —	2021 年 9 月 30 日	CodeDeploy
AWS CodePipeline	arn:aws:states:::aws-sdk:codepipeline: *[apiAction]*	2021 年 9 月 30 日	CodePipeline
AWS CodeStar	arn:aws:states:::aws-sdk:codestar: *[apiAction]*	2021 年 9 月 30 日	CodeStar

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications:` *[apiAction]*	2021 年 9 月 30 日	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections:` *[apiAction]*	2021 年 9 月 30 日	CodeStarConnections
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer:` *[apiAction]*	2021 年 9 月 30 日	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config:` *[apiAction]*	2021 年 9 月 30 日	Config
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer:` *[apiAction]*	2021 年 9 月 30 日	CostExplorer
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport:` *[apiAction]*	2021 年 9 月 30 日	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange:` *[apiAction]*	2021 年 9 月 30 日	DataExchange
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline:` *[apiAction]*	2021 年 9 月 30 日	DataPipeline

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS DataSync	`arn:aws:states:::aws-sdk:datasync:` *[apiAction]*	2021 年 9 月 30 日	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration:` *[apiAction]*	2021 年 9 月 30 日	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm:` *[apiAction]*	2021 年 9 月 30 日	DeviceFarm
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect:` *[apiAction]* *** —	2021 年 9 月 30 日	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory:` *[apiAction]*	2021 年 9 月 30 日	Directory
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect:` *[apiAction]*	2021 年 9 月 30 日	Ec2InstanceConnect
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk:` *[apiAction]*	2021 年 9 月 30 日	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive:` *[apiAction]*	2021 年 9 月 30 日	MediaLive
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage:` *[apiAction]* *** —	2021 年 9 月 30 日	MediaPackage

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod: [apiAction]`	2021 年 9 月 30 日	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore: [apiAction]`	2021 年 9 月 30 日	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis: [apiAction]`	2021 年 9 月 30 日	Fis
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms: [apiAction]`	2021 年 9 月 30 日	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue: [apiAction]`	2021 年 9 月 30 日	Glue
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew: [apiAction]`	2021 年 9 月 30 日	DataBrew
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass: [apiAction]`	2021 年 9 月 30 日	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation: [apiAction]`	2021 年 9 月 30 日	GroundStation
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam: [apiAction]`	2021 年 9 月 30 日	Iam

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS IoT	arn:aws:states:::aws-sdk:iot: *[apiAction]* ***	2021 年 9 月 30 日	Iot
AWS IoT 1-Click	arn:aws:states:::aws-sdk:iot1clickprojects: *[apiAction]*	2021 年 9 月 30 日	Iot1ClickProjects
AWS IoT Analytics	arn:aws:states:::aws-sdk:iotanalytics: *[apiAction]*	2021 年 9 月 30 日	IoTAnalytics
AWS IoT Core Device Advisor	arn:aws:states:::aws-sdk:iotdeviceadvisor: *[apiAction]* ***	2021 年 9 月 30 日	IotDeviceAdvisor
AWS IoT Events	arn:aws:states:::aws-sdk:ioteevents: *[apiAction]*	2021 年 9 月 30 日	IotEvents
AWS IoT Events 資料	arn:aws:states:::aws-sdk:ioteventsdata: *[apiAction]*	2021 年 9 月 30 日	IotEventsData
AWS IoT Fleet Hub	arn:aws:states:::aws-sdk:iotfleethub: *[apiAction]*	2021 年 9 月 30 日	IoTFleetHub
AWS IoT Greengrass Version 2	arn:aws:states:::aws-sdk:greengrassv2: *[apiAction]*	2021 年 9 月 30 日	GreengrassV2
AWS IoT 任務資料 Plane	arn:aws:states:::aws-sdk:iotjobsdataplane: *[apiAction]*	2021 年 9 月 30 日	IotJobsDataPlane

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS IoT Secure Tunneling	arn:aws:states:::aws-sdk:iotsecuretunneling: *[apiAction]*	2021 年 9 月 30 日	IoTSecureTunneling
AWS IoT SiteWise	arn:aws:states:::aws-sdk:iotsitewise: *[apiAction]*	2021 年 9 月 30 日	IoTSiteWise
AWS IoT Wireless	arn:aws:states:::aws-sdk:iotwireless: *[apiAction]*	2021 年 9 月 30 日	IotWireless
AWS Key Management Service	arn:aws:states:::aws-sdk:kms: *[apiAction]*	2021 年 9 月 30 日	Kms
AWS Lake Formation	arn:aws:states:::aws-sdk:lakeformation: *[apiAction]*	2021 年 9 月 30 日	LakeFormation
AWS Lambda	arn:aws:states:::aws-sdk:lambda: *[apiAction]* ***	2021 年 9 月 30 日	Lambda
AWS License Manager	arn:aws:states:::aws-sdk:licensemanager: *[apiAction]*	2021 年 9 月 30 日	LicenseManager
AWS Marketplace	arn:aws:states:::aws-sdk:marketplacecatalog: *[apiAction]*	2021 年 9 月 30 日	MarketplaceCatalog
AWS Marketplace Commerce Analytics	arn:aws:states:::aws-sdk:marketplacecommerceanalytics: *[apiAction]*	2021 年 9 月 30 日	MarketplaceCommerceAnalytics

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Marketplace Entitlement Service	arn:aws:states:::aws-sdk:marketplaceentitlement: *[apiAction]*	2021 年 9 月 30 日	MarketplaceEntitlement
AWS Elemental MediaTailor	arn:aws:states:::aws-sdk:mediatailor: *[apiAction]*	2021 年 9 月 30 日	MediaTailor
AWS Migration Hub	arn:aws:states:::aws-sdk:migrationhub: *[apiAction]*	2021 年 9 月 30 日	MigrationHub
AWS Migration Hub Config	arn:aws:states:::aws-sdk:migrationhubconfig: *[apiAction]*	2021 年 9 月 30 日	MigrationHubConfig
AWS Migration Hub 策略建議	arn:aws:states:::aws-sdk:migrationhubstrategy: *[apiAction]*	2022 年 4 月 19 日	MigrationHubStrategy
AWS Mobile	arn:aws:states:::aws-sdk:mobile: *[apiAction]*	2021 年 9 月 30 日	
AWS Network Firewall	arn:aws:states:::aws-sdk:networkfirewall: *[apiAction]*	2021 年 9 月 30 日	NetworkFirewall
AWS OpsWorks	arn:aws:states:::aws-sdk:opsworks: *[apiAction]*	2021 年 9 月 30 日	OpsWorks

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS OpsWorks CM	arn:aws:states:::aws-sdk:opsworkscm: *[apiAction]*	2021 年 9 月 30 日	OpsWorksCm
AWS Organizations	arn:aws:states:::aws-sdk:organizations: *[apiAction]*	2021 年 9 月 30 日	Organizations
AWS Outposts	arn:aws:states:::aws-sdk:outposts: *[apiAction]*	2021 年 9 月 30 日	Outposts
AWS Panorama	arn:aws:states:::aws-sdk:panorama: *[apiAction]*	2022 年 4 月 19 日	Panorama
Amazon Relational Database Service Performance Insights	arn:aws:states:::aws-sdk:pi: *[apiAction]*	2021 年 9 月 30 日	Pi
AWS 價格表	arn:aws:states:::aws-sdk:pricing: *[apiAction]*	2021 年 9 月 30 日	Pricing
Amazon Relational Database Service	arn:aws:states:::aws-sdk:rdsdata: *[apiAction]* ***	2021 年 9 月 30 日	RdsData
AWS Resilience Hub	arn:aws:states:::aws-sdk:resiliencehub: *[apiAction]*	2022 年 4 月 19 日	Resiliencehub
AWS Resource Access Manager	arn:aws:states:::aws-sdk:ram: *[apiAction]*	2021 年 9 月 30 日	Ram

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Resource Groups	arn:aws:states:::aws-sdk:resourcegroups: *[apiAction]*	2021 年 9 月 30 日	ResourceGroups
AWS Resource Groups Tagging API	arn:aws:states:::aws-sdk:resourcegroupstaggingapi: *[apiAction]*	2021 年 9 月 30 日	ResourceGroupsTaggingApi
AWS RoboMaker	arn:aws:states:::aws-sdk:robomaker: *[apiAction]*	2021 年 9 月 30 日	RoboMaker
AWS IAM Identity Center	arn:aws:states:::aws-sdk:identitystore: *[apiAction]*	2021 年 9 月 30 日	Identitystore
IAM Identity Center OIDC	arn:aws:states:::aws-sdk:ssooidc: *[apiAction]*	2021 年 9 月 30 日	SsoOidc
AWS Secrets Manager	arn:aws:states:::aws-sdk:secretsmanager: *[apiAction]*	2021 年 9 月 30 日	SecretsManager
AWS Security Token Service	arn:aws:states:::aws-sdk:sts: *[apiAction]* ***	2021 年 9 月 30 日	Sts
AWS Security Hub	arn:aws:states:::aws-sdk:securityhub: *[apiAction]*	2021 年 9 月 30 日	SecurityHub
AWS Server Migration Service	arn:aws:states:::aws-sdk:sms: *[apiAction]*	2021 年 9 月 30 日	Sms

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Service Catalog	arn:aws:states:::aws-sdk:servicecatalog: *[apiAction]*	2021 年 9 月 30 日	ServiceCatalog
AWS Service Catalog AppRegistry	arn:aws:states:::aws-sdk:servicecatalogappregistry: *[apiAction]*	2021 年 9 月 30 日	ServiceCatalogAppRegistry
AWS Shield	arn:aws:states:::aws-sdk:shield: *[apiAction]* ***	2021 年 9 月 30 日	Shield
AWS Signer	arn:aws:states:::aws-sdk:signer: *[apiAction]*	2021 年 9 月 30 日	Signer
IAM Identity Center	arn:aws:states:::aws-sdk:sso: *[apiAction]*	2021 年 9 月 30 日	Sso
IAM Identity Center Admin	arn:aws:states:::aws-sdk:ssoadmin: *[apiAction]*	2021 年 9 月 30 日	SsoAdmin
AWS Step Functions	arn:aws:states:::aws-sdk:sfn: *[apiAction]*	2021 年 9 月 30 日	Sfn
AWS Storage Gateway	arn:aws:states:::aws-sdk:storagegateway: *[apiAction]*	2021 年 9 月 30 日	StorageGateway
AWS Support	arn:aws:states:::aws-sdk:support: *[apiAction]*	2021 年 9 月 30 日	Support

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Systems Manager Incident Manager	`arn:aws:states:::aws-sdk:ssmincidents: [apiAction]`		SsmIncidents
AWS Transfer Family	`arn:aws:states:::aws-sdk:transfer: [apiAction]`	2021 年 9 月 30 日	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf: [apiAction]`	2021 年 9 月 30 日	Waf
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional: [apiAction]`	2021 年 9 月 30 日	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2: [apiAction]`	2021 年 9 月 30 日	Wafv2
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	2021 年 9 月 30 日	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	2021 年 9 月 30 日	XRay
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering: [apiAction]`	2021 年 9 月 30 日	MarketplaceMetering

服務名稱	Task狀態資源	支援的日期	異常前綴
AWS Serverless Application Repository	arn:aws:states:::aws-sdk:serverlessapplicationrepository: *[apiAction]*	2021 年 9 月 30 日	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	arn:aws:states:::aws-sdk:accessanalyzer: *[apiAction]*	2021 年 9 月 30 日	AccessAnalyzer
Alexa for Business	arn:aws:states:::aws-sdk:alexaforbusiness: *[apiAction]*	2021 年 9 月 30 日	AlexaForBusiness
Amazon API Gateway	arn:aws:states:::aws-sdk:apigateway: *[apiAction]*	2021 年 9 月 30 日	ApiGateway
Amazon API Gateway	arn:aws:states:::aws-sdk:apigatewayv2: *[apiAction]*	2021 年 9 月 30 日	ApiGatewayV2
Amazon AppIntegrations	arn:aws:states:::aws-sdk:appintegrations: *[apiAction]*	2021 年 9 月 30 日	AppIntegrations
Amazon AppStream 2.0	arn:aws:states:::aws-sdk:appstream: *[apiAction]*	2021 年 9 月 30 日	AppStream
Amazon AppFlow	arn:aws:states:::aws-sdk:appflow: *[apiAction]*	2021 年 9 月 30 日	Appflow

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Athena	arn:aws:states:::aws-sdk:athena: *[apiAction]*	2021 年 9 月 30 日	Athena
Amazon Augmented AI	arn:aws:states:::aws-sdk:sagemakera2iruntime: *[apiAction]*	2021 年 9 月 30 日	SageMaker A2IRuntime
Amazon Braket	arn:aws:states:::aws-sdk:braket: *[apiAction]*	2021 年 9 月 30 日	Braket
Amazon Chime	arn:aws:states:::aws-sdk:chime: *[apiAction]*	2021 年 9 月 30 日	Chime
Amazon Chime Meetings	arn:aws:states:::aws-sdk:chimesdkmeetings: *[apiAction]*	2022 年 4 月 19 日	ChimeSdkMeetings
Amazon Cloud Directory	arn:aws:states:::aws-sdk:clouddirectory: *[apiAction]*	2021 年 9 月 30 日	CloudDirectory
Amazon CloudFront	arn:aws:states:::aws-sdk:cloudfront: *[apiAction]*	2021 年 9 月 30 日	CloudFront
Amazon CloudSearch	arn:aws:states:::aws-sdk:cloudsearch: *[apiAction]*	2021 年 9 月 30 日	CloudSearch
Amazon CloudWatch	arn:aws:states:::aws-sdk:cloudwatch: *[apiAction]*	2021 年 9 月 30 日	CloudWatch

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights:` *[apiAction]*	2021 年 9 月 30 日	ApplicationInsights
CloudWatch Evidently	`arn:aws:states:::aws-sdk:evidently:` *[apiAction]*	2022 年 4 月 19 日	Evidently
Amazon CloudWatch Logs	`arn:aws:states:::aws-sdk:cloudwatchlogs:` *[apiAction]*	2021 年 9 月 30 日	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum:` *[apiAction]*	2022 年 4 月 19 日	Rum
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics:` *[apiAction]*	2021 年 9 月 30 日	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler:` *[apiAction]*	2021 年 9 月 30 日	CodeGuruProfiler
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer:` *[apiAction]*	2021 年 9 月 30 日	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity:` *[apiAction]*	2021 年 9 月 30 日	CognitoIdentity

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Cognito Identity Provider	arn:aws:states:::aws-sdk:cognitoidentityprovider: *[apiAction]*	2021 年 9 月 30 日	CognitoIdentityProvider
Amazon Cognito Sync	arn:aws:states:::aws-sdk:cognitosync: *[apiAction]*	2021 年 9 月 30 日	CognitoSync
Amazon Comprehend	arn:aws:states:::aws-sdk:comprehend: *[apiAction]*	2021 年 9 月 30 日	Comprehend
Amazon Comprehend Medical	arn:aws:states:::aws-sdk:comprehendmedical: *[apiAction]* ***	2021 年 9 月 30 日	ComprehendMedical
Amazon Connect Contact Lens	arn:aws:states:::aws-sdk:connectcontactlens: *[apiAction]*	2021 年 9 月 30 日	ConnectContactLens
Amazon Connect Participant Service	arn:aws:states:::aws-sdk:connectparticipant: *[apiAction]*	2021 年 9 月 30 日	ConnectParticipant
Amazon Connect	arn:aws:states:::aws-sdk:connect: *[apiAction]*	2021 年 9 月 30 日	Connect
Amazon Connect Voice ID	arn:aws:states:::aws-sdk:voiceid: *[apiAction]*	2022 年 4 月 19 日	VoiceId
Amazon Connect Wisdom	arn:aws:states:::aws-sdk:wisdom: *[apiAction]*	2022 年 4 月 19 日	Wisdom

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Data Lifecycle Manager	arn:aws:states:::aws-sdk:dlm: *[apiAction]*	2021 年 9 月 30 日	Dlm
Amazon Detective	arn:aws:states:::aws-sdk:detective: *[apiAction]*	2021 年 9 月 30 日	Detective
Amazon DevOps Guru	arn:aws:states:::aws-sdk:devopsguru: *[apiAction]*	2021 年 9 月 30 日	DevOpsGuru
Amazon DocumentDB (with MongoDB compatibility)	arn:aws:states:::aws-sdk:docdb: *[apiAction]*	2021 年 9 月 30 日	DocDb
Amazon DynamoDB	arn:aws:states:::aws-sdk:dynamodb: *[apiAction]*	2021 年 9 月 30 日	DynamoDb
Amazon DynamoDB Streams	arn:aws:states:::aws-sdk:dynamodbstreams: *[apiAction]*	2021 年 9 月 30 日	DynamoDbStreams
Amazon EC2 Container Registry	arn:aws:states:::aws-sdk:ecr: *[apiAction]*	2021 年 9 月 30 日	Ecr
Amazon EC2 Container Service	arn:aws:states:::aws-sdk:ecs: *[apiAction]*	2021 年 9 月 30 日	Ecs
Amazon EC2 Systems Manager	arn:aws:states:::aws-sdk:ssm: *[apiAction]*	2021 年 9 月 30 日	Ssm

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers:` *[apiAction]* ***	2021 年 9 月 30 日	EmrContainers
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache:` *[apiAction]*	2021 年 9 月 30 日	ElastiCache
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference:` *[apiAction]*	2021 年 9 月 30 日	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs:` *[apiAction]*	2021 年 9 月 30 日	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2:` *[apiAction]*	2021 年 9 月 30 日	Ec2
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic:` *[apiAction]*	2021 年 9 月 30 日	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs:` *[apiAction]* ***	2021 年 9 月 30 日	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks:` *[apiAction]*	2021 年 9 月 30 日	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr:` *[apiAction]*	2021 年 9 月 30 日	Emr

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder: [apiAction]` ***	2021 年 9 月 30 日	ElasticTranscoder
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch: [apiAction]`	2021 年 9 月 30 日	Elasticsearch
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:opensearch: [apiAction]`	2022 年 4 月 19 日	OpenSearch
Amazon EventBridge	`arn:aws:states:::aws-sdk:eventbridge: [apiAction]`	2021 年 9 月 30 日	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx: [apiAction]`	2021 年 9 月 30 日	FSx
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery: [apiAction]`	2021 年 9 月 30 日	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast: [apiAction]`	2021 年 9 月 30 日	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector: [apiAction]`	2021 年 9 月 30 日	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift: [apiAction]`	2021 年 9 月 30 日	Amazon GameLift

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon GameSparks	arn:aws:states:::aws-sdk:gamesparks: *[apiAction]*	2022 年 7 月 27 日	GameSparks
Amazon S3 Glacier	arn:aws:states:::aws-sdk:glacier: *[apiAction]*	2021 年 9 月 30 日	Glacier
Amazon GuardDuty	arn:aws:states:::aws-sdk:guardduty: *[apiAction]*	2021 年 9 月 30 日	GuardDuty
AWS HealthLake	arn:aws:states:::aws-sdk:healthlake: *[apiAction]*	2021 年 9 月 30 日	HealthLake
Amazon Honeycode	arn:aws:states:::aws-sdk:honeycode: *[apiAction]*	2021 年 9 月 30 日	Honeycode
Amazon Inspector	arn:aws:states:::aws-sdk:inspector: *[apiAction]*	2021 年 9 月 30 日	Inspector
Amazon Inspector V2	arn:aws:states:::aws-sdk:inspector2: *[apiAction]*	2022 年 4 月 19 日	Inspector2
Amazon Interactive Video Service	arn:aws:states:::aws-sdk:ivs: *[apiAction]*	2021 年 9 月 30 日	Ivs
Amazon Kendra	arn:aws:states:::aws-sdk:kendra: *[apiAction]*	2021 年 9 月 30 日	Kendra

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Kinesis	arn:aws:states:::aws-sdk:kinesis: *[apiAction]* ***	2021 年 9 月 30 日	Kinesis
Amazon Kinesis Analytics	arn:aws:states:::aws-sdk:kinesisanalytics: *[apiAction]*	2021 年 9 月 30 日	KinesisAnalytics
Amazon Kinesis Analytics V2	arn:aws:states:::aws-sdk:kinesisanalyticsv2: *[apiAction]*	2021 年 9 月 30 日	KinesisAnalyticsV2
Amazon Kinesis Firehose	arn:aws:states:::aws-sdk:firehose: *[apiAction]*	2021 年 9 月 30 日	Firehose
Amazon Kinesis Video Signaling Channels	arn:aws:states:::aws-sdk:kinesisvideosignaling: *[apiAction]*	2021 年 9 月 30 日	KinesisVideoSignaling
Amazon Kinesis Video Streams	arn:aws:states:::aws-sdk:kinesisvideo: *[apiAction]*	2021 年 9 月 30 日	KinesisVideo
Amazon Kinesis Video Streams Archived Media	arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia: *[apiAction]*	2021 年 9 月 30 日	KinesisVideoArchivedMedia
Amazon Kinesis video stream	arn:aws:states:::aws-sdk:kinesisvideomedia: *[apiAction]*	2021 年 9 月 30 日	KinesisVideoMedia

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Lex Model Building Service	arn:aws:states:::aws-sdk:lexmodelbuilding: *[apiAction]*	2021 年 9 月 30 日	LexModelBuilding
Amazon Lex Model Building Service V2	arn:aws:states:::aws-sdk:lexmodelsv2: *[apiAction]*	2021 年 9 月 30 日	LexModelsV2
Amazon Lex	arn:aws:states:::aws-sdk:lexruntime: *[apiAction]*	2021 年 9 月 30 日	LexRuntime
Amazon Lex Runtime V2	arn:aws:states:::aws-sdk:lexruntimev2: *[apiAction]* ***	2021 年 9 月 30 日	LexRuntimeV2
Amazon Lightsail	arn:aws:states:::aws-sdk:lightsail: *[apiAction]*	2021 年 9 月 30 日	Lightsail
Amazon Location Service	arn:aws:states:::aws-sdk:location: *[apiAction]*	2021 年 9 月 30 日	Location
Amazon Lookout for Equipment	arn:aws::states:::aws-sdk:lookoutequipment: *[apiAction]*	2021 年 9 月 30 日	LookoutEquipment
Amazon Lookout for Metrics	arn:aws:states:::aws-sdk:lookoutmetrics: *[apiAction]*	2021 年 9 月 30 日	LookoutMetrics
Amazon Lookout for Vision	arn:aws:states:::aws-sdk:lookoutvision: *[apiAction]*	2021 年 9 月 30 日	LookoutVision

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon MQ	arn:aws:states:::aws-sdk:mq:*[apiAction]*	2021 年 9 月 30 日	Mq
Amazon Macie	arn:aws:states:::aws-sdk:macie:*[apiAction]*	2021 年 9 月 30 日	
Amazon Macie 2	arn:aws:states:::aws-sdk:macie2:*[apiAction]*	2021 年 9 月 30 日	Macie2
Amazon Managed Blockchain	arn:aws:states:::aws-sdk:managedblockchain:*[apiAction]*	2021 年 9 月 30 日	ManagedBlockchain
Amazon Managed Grafana	arn:aws:states:::aws-sdk:grafana:*[apiAction]*	2022 年 4 月 19 日	Grafana
Amazon Managed Service for Prometheus	arn:aws:states:::aws-sdk:amp:*[apiAction]*	2021 年 9 月 30 日	Amp
Amazon Managed Streaming for Apache Kafka	arn:aws:states:::aws-sdk:kafka:*[apiAction]*	2021 年 9 月 30 日	Kafka
Amazon MSK Connect	arn:aws:states:::aws-sdk:kafkaconnect:*[apiAction]*	2022 年 4 月 19 日	KafkaConnect
Amazon Managed Workflows for Apache Airflow	arn:aws:states:::aws-sdk:mwaa:*[apiAction]*	2021 年 9 月 30 日	Mwaa

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk: [apiAction]`	2021 年 9 月 30 日	MTurk
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb: [apiAction]`	2022 年 4 月 19 日	MemoryDB
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble: [apiAction]`	2021 年 9 月 30 日	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize: [apiAction]`	2021 年 9 月 30 日	Personalize
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents: [apiAction]`	2021 年 9 月 30 日	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime: [apiAction]`	2021 年 9 月 30 日	PersonalizeRuntime
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint: [apiAction]`	2021 年 9 月 30 日	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail: [apiAction]`	2021 年 9 月 30 日	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice: [apiAction]`	2021 年 9 月 30 日	PinpointSmsVoice

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Pinpoint SMS and Voice V2 Service	arn:aws:states:::aws-sdk:pinpointsmsvoicev2: *[apiAction]*	2022 年 7 月 27 日	PinpointSmsVoiceV2
Amazon Polly	arn:aws:states:::aws-sdk:polly: *[apiAction]*	2021 年 9 月 30 日	Polly
Amazon QLDB	arn:aws:states:::aws-sdk:qldb: *[apiAction]*	2021 年 9 月 30 日	Qldb
Amazon QLDB Session	arn:aws:states:::aws-sdk:qldbsession: *[apiAction]*	2021 年 9 月 30 日	QldbSession
Amazon QuickSight	arn:aws:states:::aws-sdk:quicksight: *[apiAction]*	2021 年 9 月 30 日	QuickSight
Amazon Redshift	arn:aws:states:::aws-sdk:redshift: *[apiAction]*	2021 年 9 月 30 日	Redshift
Amazon Redshift Data API	arn:aws:states:::aws-sdk:redshiftdata: *[apiAction]*	2021 年 9 月 30 日	RedshiftData
Amazon Rekognition	arn:aws:states:::aws-sdk:rekognition: *[apiAction]*	2021 年 9 月 30 日	Rekognition
Amazon Relational Database Service	arn:aws:states:::aws-sdk:rds: *[apiAction]*	2021 年 9 月 30 日	Rds

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Route 53	arn:aws:states:::aws-sdk:route53: *[apiAction]*	2021 年 9 月 30 日	Route53
Amazon Route 53 Recovery Control Config	arn:aws:states:::aws-sdk:route53recoverycontrolconfig: *[apiAction]*	2021 年 9 月 30 日	Route53RecoveryControlConfig
Amazon Route 53 Domains	arn:aws:states:::aws-sdk:route53domains: *[apiAction]*	2021 年 9 月 30 日	Route53Domains
Amazon Route 53 Resolver	arn:aws:states:::aws-sdk:route53resolver: *[apiAction]*	2021 年 9 月 30 日	Route53Resolver
Amazon S3 on Outposts	arn:aws:states:::aws-sdk:s3outposts: *[apiAction]*	2021 年 9 月 30 日	S3Outposts
Amazon SageMaker Runtime Feature Store Runtime	arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime: *[apiAction]*	2021 年 9 月 30 日	SageMakerRuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	arn:aws:states:::aws-sdk:sagemakerruntime: *[apiAction]*	2021 年 9 月 30 日	SageMakerRuntimeRuntime
Amazon SageMaker	arn:aws:states:::aws-sdk:sagemaker: *[apiAction]*	2021 年 9 月 30 日	SageMakerRuntime

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge:` *[apiAction]*	2021 年 9 月 30 日	SagemakerEdge
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses:` *[apiAction]*	2021 年 9 月 30 日	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2:` *[apiAction]*	2021 年 9 月 30 日	SesV2
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *[apiAction]*	2021 年 9 月 30 日	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *[apiAction]*	2021 年 9 月 30 日	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *[apiAction]* ***	2021 年 9 月 30 日	S3
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *[apiAction]*	2021 年 9 月 30 日	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *[apiAction]*	2021 年 9 月 30 日	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *[apiAction]*	2021 年 9 月 30 日	Transcribe

服務名稱	Task狀態資源	支援的日期	異常前綴
Amazon Translate	arn:aws:states:::aws-sdk:translate: *[apiAction]*	2021 年 9 月 30 日	Translate
Amazon WorkDocs	arn:aws:states:::aws-sdk:workdocs: *[apiAction]*	2021 年 9 月 30 日	WorkDocs
Amazon WorkMail	arn:aws:states:::aws-sdk:workmail: *[apiAction]*	2021 年 9 月 30 日	WorkMail
Amazon WorkMail Message Flow	arn:aws:states:::aws-sdk:workmailmessageflow: *[apiAction]*	2021 年 9 月 30 日	WorkMailMessageFlow
Amazon WorkSpaces	arn:aws:states:::aws-sdk:workspaces: *[apiAction]*	2021 年 9 月 30 日	WorkSpaces
Amazon WorkSpaces Web	arn:aws:states:::aws-sdk:workspacesweb: *[apiAction]*	2022 年 4 月 19 日	WorkSpacesWeb
Amplify	arn:aws:states:::aws-sdk:amplifybackend: *[apiAction]*	2021 年 9 月 30 日	AmplifyBackend
Amplify UI Builder	arn:aws:states:::aws-sdk:amplifyuibuilder: *[apiAction]*	2022 年 4 月 19 日	AmplifyUiBuilder

服務名稱	Task狀態資源	支援的日期	異常前綴
Application Auto Scaling	arn:aws:states:::aws-sdk:applicationautoscaling: *[apiAction]*	2021 年 9 月 30 日	ApplicationAutoScaling
Amazon EC2 Auto Scaling	arn:aws:states:::aws-sdk:autoscaling: *[apiAction]*	2021 年 9 月 30 日	Auto Scaling
CodeArtifact	arn:aws:states:::aws-sdk:codeartifact: *[apiAction]*	2021 年 9 月 30 日	Codeartifact
DynamoDB Accelerator	arn:aws:states:::aws-sdk:dax: *[apiAction]*	2021 年 9 月 30 日	Dax
EC2 Image Builder	arn:aws:states:::aws-sdk:imagebuilder: *[apiAction]*	2021 年 9 月 30 日	Imagebuilder
AWS Elastic Disaster Recovery	arn:aws:states:::aws-sdk:drs: *[apiAction]*	2022 年 4 月 19 日	Drs
Elastic Load Balancing	arn:aws:states:::aws-sdk:elasticloadbalancing: *[apiAction]*	2021 年 9 月 30 日	ElasticLoadBalancing
Elastic Load Balancing V2	arn:aws:states:::aws-sdk:elasticloadbalancingv2: *[apiAction]*	2021 年 9 月 30 日	ElasticLoadBalancingV2

服務名稱	Task狀態資源	支援的日期	異常前綴
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect: [apiAction]`	2021 年 9 月 30 日	MediaConnect
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control: [apiAction]` ***	2021 年 9 月 30 日	S3Control
Recycle Bin for Amazon EBS	`arn:aws:states:::aws-sdk:rbin: [apiAction]`	2022 年 4 月 19 日	Rbin
Savings Plans	`arn:aws:states:::aws-sdk:savingsplans: [apiAction]`	2021 年 9 月 30 日	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas: [apiAction]`	2021 年 9 月 30 日	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas: [apiAction]`	2021 年 9 月 30 日	ServiceQuotas
AWS Snowball	`arn:aws:states:::aws-sdk:snowball: [apiAction]`	2021 年 9 月 30 日	Snowball

支援服務不支援的 API 動作

下表列出 AWS SDK 服務整合不支援的 API 動作。右欄包含左欄中列出的服務目前不支援的 API 動作。

服務名稱	不支援的 API 動作
AWS Application Discovery Service	• DescribeExportConfigurations • ExportConfigurations
Amazon Bedrock	• InvokeModelWithResponseStream
Amazon 基岩運行時的代理	• InvokeAgent
AWS CodeDeploy	• BatchGetDeploymentInstances • GetDeploymentInstance • ListDeploymentInstances • SkipWaitTimeForInstanceTermination
Amazon Comprehend Medical	• DetectEntities
AWS Direct Connect	• AllocateConnectionOnInterconnect • DescribeConnectionLoa • DescribeConnectionsOnInterconnect • DescribeInterconnectLoa
Amazon Elastic File System	• CreateTags
Amazon Elastic Transcoder	• TestRole
Amazon EMR	• DescribeJobFlows
AWS IoT	• AttachPrincipalPolicy • ListPrincipalPolicies • DetachPrincipalPolicy • ListPolicyPrincipals • DetachPrincipalPolicy

服務名稱	不支援的 API 動作
AWS IoT 核心設備顧問	• `ListTestCases`
Amazon Kinesis	• `SubscribeToShard`
AWS Lambda	• `InvokeAsync` • `InvokeWithResponseStream`
Amazon Lex 運行時	• `StartConversation`
AWS Elemental MediaPackage	• `RotateChannelCredentials`
Amazon Relational Database Service	• `ExecuteSql`
Amazon Simple Storage Service	• `SelectObjectContent`
Amazon S3 控制	• `SelectObjectContent`
AWS Shield	• `DeleteSubscription`
AWS Security Token Service	• `AssumeRole` • `AssumeRoleWithSAML` • `AssumeRoleWithWebIdentity`

已淘汰的 AWS SDK 服務整合

下列 AWS SDK 服務整合現已淘汰：

- AWS 移動
- Amazon Macie

Step Functions 的最佳化整合

下列主題包括 Amazon 州語言中支援的 API、參數和請求/回應語法，以協調其他服務。 AWS 這些主題也提供範例程式碼。您可以直接從州`Resource`欄位中的 Amazon `Task` 州語言呼叫最佳化整合服務。

您可以使用三種服務整合模式：

- 請求響應（默認）-等待 HTTP 響應，然後轉到下一個狀態
- 執行 Job (.sync)-等待工作完成
- 等待回調（.waitForTaskToken）-暫停工作流程，直到返回任務令牌

標準工作流程和 Express 工作流程支援相同的整合，但不支援相同的整合模式。

- 每個集成的優化集成模式支持都不同。
- Express Job 流程不支援執行工作 (.sync) 或等待回呼 (. waitForTask令牌）。
- 如需詳細資訊，請參閱 標準與快速工作流程。

Standard Workflows

支援的服務整合

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
最佳化整合	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS SDK 整合	超過二百	✓		✓

Express Workflows

支援的服務整合

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
最佳化整合	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		

	服務	請求回應	執行工作 (.sync)	等候回呼 (.waitForTaskToken)
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS SDK 整合	超過二百	✓		

使用 Step Functions 呼叫 API Gateway

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> 最佳化 API Gateway 整合與 API Gateway AWS SDK 整合有何不同
>
> - `apigateway:invoke`:在 AWS SDK 服務集成中沒有相同的內容。而是，「最佳化 API Gateway」服務會直接呼叫您的 API Gateway 端點。

您可以使用 Amazon API Gateway 來建立、發佈、維護和監控 HTTP 和 REST API。若要與 API Gateway 整合，您可以在 Step Functions 式中定義一個`Task`狀態，該狀態可直接呼叫 API Gateway HTTP 或 API Gateway REST 端點，而無需撰寫程式碼或依賴其他基礎結構。`Task`狀態定義包含 API 呼叫的所有必要資訊。您也可以選擇不同的授權方法。

> Note
>
> Step Functions 支援透過 API Gateway 呼叫 HTTP 端點的功能，但目前不支援呼叫一般 HTTP 端點的功能。

API Gateway 功能支援

Step Functions API Gateway 整合支援部分，但並非所有 API Gateway 功能。如需支援功能的詳細清單，請參閱下列內容。

- 由 Step Functions API Gateway REST API 和 API Gateway HTTP API 整合提供支援：
  - 授權者：IAM（使用簽名版本 4），無身份驗證，Lambda 授權者（基於請求參數和基於令牌的自定義標題）
  - API 類型：區域
  - API 管理：API Gateway API 網域名稱、API 階段、路徑、查詢參數、要求主體
- 由 Step Functions API Gateway HTTP API 整合提供支援。不支援提供邊緣最佳化 API 選項的 Step Functions 式 API Gateway REST API 整合。
- Step Functions API Gateway 整合不支援：
  - 授權者：Amazon Cognito，本機開放 ID Connect/OAuth 2.0，基於令牌的 Lambda 授權者的授權標頭

- API 類型：私有
- API 管理：自訂網域名稱

如需 API Gateway 及其 HTTP 和 REST API 的詳細資訊，請參閱下列內容。

- Amazon API Gateway 概念頁面。
- 在 API Gateway 開發人員指南中選擇 HTTP API 和其餘 API。

要求格式

當您建立Task狀態定義時，Step Functions 會驗證參數、建立必要的 URL 來執行呼叫，然後呼叫 API。響應包括 HTTP 狀態碼，標頭和響應主體。請求格式同時具有必要和可選參數。

必要的請求參數

- `ApiEndpoint`
  - 類型：`String`
  - API Gateway 網址的主機名稱。格式是 *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`。

    API ID 只能包含下列字母數字字元的組合：`0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - 類型：`Enum`
  - HTTP 方法，它必須是下列其中一種：
    - `GET`
    - `POST`
    - `PUT`
    - `DELETE`
    - `PATCH`
    - `HEAD`
    - `OPTIONS`

可選的請求參數

- Headers
  - 類型：JSON
  - HTTP 標頭允許與同一個密鑰關聯的值的列表。
- Stage
  - 類型：String
  - 在 API Gateway 中部署 API 的階段名稱。對於使用該$default階段的任何 HTTP API 來說，它都是可選的。
- Path
  - 類型：String
  - 附加在 API 端點之後的路徑參數。
- QueryParameters
  - 類型：JSON
  - 查詢字串只允許與相同索引鍵相關聯的值清單。
- RequestBody
  - 類型：JSON 或 String。
  - HTTP 要求主體。它的類型可以是一個JSON對象或String。 RequestBody僅支援PATCHPOST、和 PUT HTTP 方法。
- AllowNullValues
  - 類型：BOOLEAN
  - 設置AllowNullValues為true將允許您傳遞空值，如下所示：

```
{
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "name": null
    }
}
```

- AuthType
  - 類型：JSON
  - 驗證方法。預設方法為NO_AUTH。允許的值為：

- NO_AUTH
- IAM_ROLE
- RESOURCE_POLICY

如需詳細資訊，請參閱驗證和授權。

> **Note**
>
> 基於安全性考量，目前不允許使用下列 HTTP 標頭金鑰：
>
> - 任何以X-Forwarded、X-Amz或X-Amzn為前綴的項目。
> - Authorization
> - Connection
> - Content-md5
> - Expect
> - Host
> - Max-Forwards
> - Proxy-Authenticate
> - Server
> - TE
> - Transfer-Encoding
> - Trailer
> - Upgrade
> - Via
> - Www-Authenticate

下列程式碼範例示範如何使用 Step Functions 式叫用 API Gateway。

```
{
    "Type": "Task",
    "Resource":"arn:aws:states:::apigateway:invoke",
    "Parameters": {
        "ApiEndpoint": "example.execute-api.us-east-1.amazonaws.com",
```

```
        "Method": "GET",
        "Headers": {
            "key": ["value1", "value2"]
        },
        "Stage": "prod",
        "Path": "bills",
        "QueryParameters": {
            "billId": ["123456"]
        },
        "RequestBody": {},
        "AuthType": "NO_AUTH"
    }
}
```

身份驗證和授權

您可以使用下列驗證方法：

- 無授權：無需授權方法即可直接呼叫 API。
- IAM 角色：使用此方法，Step Functions 擔任狀態機器的角色，使用簽名版本 4（Sigv4）簽署請求，然後調用 API。
- 資源策略：Step Functions 驗證請求，然後調用 API。您必須將資源策略附加到指定以下內容的 API：
  1. 將叫用 API Gateway 的狀態機器。

     > **Important**
     >
     > 您必須指定您的狀態機器以限制對其的存取。如果您不這樣做，則任何狀態機器會使用 API 的資源策略驗證來驗證其 API Gateway 請求，將被授與存取權。

  2. 該 Step Functions 是調用 API Gateway 的服務：`"Service": "states.amazonaws.com"`。
  3. 您要存取的資源，包括：
     - 該*##*。
     - 指定區域中的*## ID*。
     - *API ###*。
     - *####*。
     - 在 *HTTP-##*（方法）。

- 的 *resource-path-specifier*.

如需資源政策範例，請參閱 Step Functions 和 API Gateway 的 IAM 政策。

如需有關資源格式的詳細資訊，請參閱 API Gateway 開發人員指南中的 API Gateway 執行 API 權限的資源格式。

> **Note**
>
> 資源策略僅支援其餘 API。

服務整合模式

API Gateway 整合支援兩種服務整合模式：

- 請求回應，這是默認的集成模式。它可以讓 Step Functions 進展到接收 HTTP 響應後立即下一步。
- 等候傳回任務字符的回呼(.waitForTaskToken)，會等到傳回包含有效負載的工作 Token 為止。要使用該.waitForTaskToken模式，請附加。 waitForTask標記到任務定義的「資源」字段末尾，如以下示例所示：

```
{
    "Type": "Task",
    "Resource":"arn:aws:states:::apigateway:invoke.waitForTaskToken",
    "Parameters": {
        "ApiEndpoint": "example.execute-api.us-east-1.amazonaws.com",
        "Method": "POST",
        "Headers": {
            "TaskToken.$": "States.Array($$.Task.Token)"
        },
        "Stage": "prod",
        "Path": "bills/add",
        "QueryParameters": {},
        "RequestBody": {
            "billId": "my-new-bill"
        },
        "AuthType": "IAM_ROLE"
    }
}
```

輸出格式

提供了以下輸出參數：

名稱	Type	描述
ResponseBody	JSON 或 String	API 呼叫的回應主體。
Headers	JSON	響應頭。
StatusCode	Integer	回應的 HTTP 狀態碼。
StatusText	String	響應的狀態文本。

一個示例響應：

```
{
    "ResponseBody": {
        "myBills": []
    },
    "Headers": {
        "key": ["value1", "value2"]
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

錯誤處理

發生錯誤時，會傳回 error ANDcause，如下所示：

- 如果 HTTP 狀態碼可用，則會以格式傳回錯誤ApiGateway.*<HTTP Status Code>*。
- 如果 HTTP 狀態碼不可用，則會以格式傳回錯誤ApiGateway.*<Exception>*。

在這兩種情況下，cause都會以字串的形式傳回。

下列範例顯示發生錯誤的回應：

```
{
```

```
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> 的狀態碼2XX表示成功，且不會傳回任何錯誤。所有其他狀態碼或拋出的異常將導致錯誤。

如需詳細資訊，請參閱:

- API Gateway 開發人員指南中的 Amazon API Gateway 概念。
- Amazon API Gateway 的 IAM 政策
- 範例專案，展示如何 呼叫 API Gateway

API Gateway 開發人員指南中的 Amazon API Gateway 概念。

使用 Step Functions 呼叫 Athena

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> **最佳化的 Athena 整合與 Athena AWS SDK 整合有何不同**
>
> - 支援執行任務 (.sync)整合模式。
> - 請求回應整合模式沒有最佳化。
> - 不支援等候傳回任務字符的回呼整合模式。

與 Amazon Athena 的 AWS Step Functions 服務整合可讓您使用 Step Functions 來啟動和停止查詢執行，以及取得查詢結果。使用 Step Functions 數，您可以執行臨機操作或排程的資料查詢，以及擷取以 S3 資料湖為目標的結果。Athena 沒有伺服器，所以不需設定和管理基礎設施，而且您只需支付執行的查詢費用。

若要 AWS Step Functions 與 Amazon Athena 整合，您可以使用提供的 Athena 服務整合 API。

服務整合 API 與對應的 Athena API 相同。並非所有 API 都支援所有整合模式，如下表所示：

API	請求回應	執行任務 (.sync)
StartQueryExecution	✓	✓
StopQueryExecution	✓	
GetQueryExecution	✓	
GetQueryResults	✓	

支援的 Amazon Athena API：

> **Note**
>
> 「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

- StartQueryExecution
  - 請求語法
  - 支援的參數：
    - ClientRequestToken
    - ExecutionParameters
    - QueryExecutionContext
    - QueryString
    - ResultConfiguration
    - WorkGroup
  - Response syntax
- StopQueryExecution
  - 請求語法
  - 支援的參數：
    - QueryExecutionId
- GetQueryExecution

  - 請求語法
  - 支援的參數：
    - QueryExecutionId
  - Response syntax
- GetQueryResults
  - 請求語法
  - 支援的參數：
    - MaxResults
    - NextToken
    - QueryExecutionId
  - Response syntax

以下內容包含啟動 Athena 查詢的工作狀態。

```
"Start an Athena query": {
  "Type": "Task",
  "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
  "Parameters": {
    "QueryString": "SELECT * FROM \"myDatabase\".\"myTable\" limit 1",
    "WorkGroup": "primary",
    "ResultConfiguration": {
       "OutputLocation": "s3://athenaQueryResult"
    }
  },
  "Next": "Get results of the query"
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

AWS Batch 使用 Step Functions 管理

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

最佳化 AWS Batch 整合與 AWS BatchAWS SDK 整合有何不同

- 整執行任務 (.sync)合模式可用。

請注意，請求回應或等候傳回任務字符的回呼整合模式沒有最佳化。

支援的 AWS Batch API：

- SubmitJob
  - 請求語法
  - 支援的參數：
    - ArrayProperties
    - ContainerOverrides
    - DependsOn
    - JobDefinition
    - JobName
    - JobQueue
    - Parameters
    - RetryStrategy
    - Timeout
    - Tags
  - 回應語法

Note

中的參數Step Functions會在中表示 PascalCase，即使原生服務 API 位於 camelCase 中。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

以下內容包含送出 AWS Batch 工作並等待工作完成的`Task`狀態。

```
{
```

```
  "StartAt": "BATCH_JOB",
  "States": {
    "BATCH_JOB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobDefinition": "preprocessing",
        "JobName": "PreprocessingBatchJob",
        "JobQueue": "SecondaryQueue",
        "Parameters.$": "$.batchjob.parameters",
        "ContainerOverrides": {
          "ResourceRequirements": [
            {
              "Type": "VCPU",
              "Value": "4"
            }
          ]
        }
      },
      "End": true
    }
  }
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 呼叫 Amazon Bedrock

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

主題

- Amazon Bedrock服務整合 API
- Amazon Bedrock整合的工作狀態定義

Amazon Bedrock服務整合 API

若要AWS Step Functions與整合Amazon Bedrock，您可以使用下列 API。這些 API 類似於對應的 Amazon Bedrock API，在傳遞的請求欄位中有些不同。

下表描述每個服務整合 API 與其對應 Amazon Bedrock API 之間的差異：

Amazon Bedrock服務整合 API 和對應的 Amazon Bedrock API

Amazon Bedrock服務整合 API	對應的 Amazon Bedrock API	差異
InvokeModel 叫用指定的Amazon Bedrock模型，使用您在要求主體中提供的輸入來執行推論。您可InvokeModel 以用來執行文字模型、影像模型和嵌入模型的推論。	InvokeModel	Amazon Bedrock服務整合 API 要求主體包含下列其他參數。 • Body— 以內容類型要求標頭中指定的格式指定輸入資料。 Body包含特定於目標模型的參數。 如果您使用 InvokeModel API，則必須指定Body參數。 Step Functions不驗證您提供的輸入Body。 Body使用Amazon Bedrock最佳化整合指定時，您可以指定最大 256 KB 的承載。如果您的承載超過 256 KB，建議您使用Input。 • Input— 指定要從中擷取輸入資料的來源。此選用欄位專用於與之Amazon Bedrock最佳化的整合Step Functions。在此欄位中，您可以指定S3Uri. 您可以在「參數」Body 中指定或指定Input，但不能同時指定兩者。 如果Input未指定指定ContentType ，則輸入資料來源的內容類型會成為的值ContentType 。

Amazon Bedrock服務整合 API	對應的 Amazon Bedrock API	差異
		• `Output`— 指定寫入 API 回應的目的地。此選用欄位專用於與之Amazon Bedrock最佳化的整合Step Functions 。在此欄位中，您可以指定`S3Uri`. 如果您指定此欄位，API 回應主體會取代為原始輸出Amazon S3位置的參照。 下面的例子顯示了用於Amazon Bedrock集成 InvokeModel API 的語法。 `{ "ModelId": String, // required "Accept": String, // default: application/json "ContentType": String, // default: application/json "Input": { // not from Bedrock API "S3Uri": String }, "Output": { // not from Bedrock API "S3Uri": String } }`
CreateModelCustomizationJob 建立微調工作以自訂基本模型。	CreateModelCustomizationJob	無

Amazon Bedrock服務整合 API	對應的 Amazon Bedrock API	差異
CreateModelCustomizationJob. 同步。 建立微調工作以自訂基本模型。	CreateModelCustomizationJob	無

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

Amazon Bedrock整合的工作狀態定義

以下任務狀態定義顯示了如何在狀態機器Amazon Bedrock中進行集成。此範例顯示 Task 狀態，`result_one`該狀態會擷取路徑指定之模型呼叫的完整結果。這是以基礎模型的推論參數為基礎。這個例子使用 Cohere 命令大語言模型（LLM）。

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::bedrock:invokeModel",
  "Parameters": {
    "ModelId": "cohere.command-text-v14",
    "Body": {
      "prompt.$": "$.prompt_one",
      "max_tokens": 250
    },
    "ContentType": "application/json",
    "Accept": "*/*"
  },
  "ResultPath": "$.result_one",
  "ResultSelector": {
    "result_one.$": "$.Body.generations[0].text"
  },
  "End": true
}
```

> **Tip**
>
> 若要部署與Amazon Bedrock您整合的狀態機器的範例 AWS 帳戶，請參閱使用執行 AI 提示鏈結 Amazon Bedrock。

AWS CodeBuild 使用 Step Functions 調用

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> **最佳化 CodeBuild 整合與 CodeBuild AWS SDK 整合有何不同**
>
> - 支援執行任務 (.sync)整合模式。
> - 在您呼叫`StopBuild`或之後`StopBuildBatch`，建置或建置批次不會立即刪除，直到某些內部工作完成以完成建置或組建的狀態 CodeBuild 為止。如果您嘗試使用`BatchDeleteBuilds`或`DeleteBuildBatch`在此期間，可能不會刪除組建或建置批次。優化的服務集成，`BatchDeleteBuilds`並`DeleteBuildBatch`包括內部重試，以簡化停止後立即刪除的用例。

與 AWS Step Functions 服務整合 AWS CodeBuild 可讓您使用 Step Functions 觸發、停止和管理組建，以及共用組建報告。使用 Step Functions，您可以設計並執行持續整合管道，以驗證應用程式的軟體變更。

並非所有 API 都支援所有整合模式，如下表所示：

API	請求回應	執行任務 (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓

API	請求回應	執行任務 (.sync)
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

> **Note**
>
> 中的參數Step Functions會以表示 PascalCase，即使原生服務 API 位於 camelCase 中。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

支援的 CodeBuild API 和語法：

- `StartBuild`
  - 請求語法
  - 支援的參數：
    - `ProjectName`
    - `ArtifactsOverride`
    - `BuildspecOverride`
    - `CacheOverride`
    - `CertificateOverride`
    - `ComputeTypeOverride`
    - `EncryptionKeyOverride`
    - `EnvironmentTypeOverride`
    - `EnvironmentVariablesOverride`
    - `GitCloneDepthOverride`
    - `GitSubmodulesConfigOverride`
    - `IdempotencyToken`
    - `ImageOverride`

    - ImagePullCredentialsTypeOverride
    - InsecureSslOverride
    - LogsConfigOverride
    - PrivilegedModeOverride
    - QueuedTimeoutInMinutesOverride
    - RegistryCredentialOverride
    - ReportBuildStatusOverride
    - SecondaryArtifactsOverride
    - SecondarySourcesOverride
    - SecondarySourcesVersionOverride
    - ServiceRoleOverride
    - SourceAuthOverride
    - SourceLocationOverride
    - SourceTypeOverride
    - SourceVersion
    - TimeoutInMinutesOverride
  - 回應語法
- StopBuild
  - 請求語法
  - 支援的參數：
    - Id
  - 回應語法
- BatchDeleteBuilds
  - 請求語法
  - 支援的參數：
    - Ids
  - 回應語法
- BatchGetReports
  - 請求語法
  - 支援的參數：

    - ReportArns
  - 回應語法
- StartBuildBatch
  - 請求語法
  - 支援的參數：
    - ProjectName
    - ArtifactsOverride
    - BuildBatchConfigOverride
    - BuildspecOverride
    - BuildTimeoutInMinutesOverride
    - CacheOverride
    - CertificateOverride
    - ComputeTypeOverride
    - DebugSessionEnabled
    - EncryptionKeyOverride
    - EnvironmentTypeOverride
    - EnvironmentVariablesOverride
    - GitCloneDepthOverride
    - GitSubmodulesConfigOverride
    - IdempotencyToken
    - ImageOverride
    - ImagePullCredentialsTypeOverride
    - InsecureSslOverride
    - LogsConfigOverride
    - PrivilegedModeOverride
    - QueuedTimeoutInMinutesOverride
    - RegistryCredentialOverride
    - ReportBuildBatchStatusOverride
    - SecondaryArtifactsOverride
    - SecondarySourcesOverride

    - SecondarySourcesVersionOverride
    - ServiceRoleOverride
    - SourceAuthOverride
    - SourceLocationOverride
    - SourceTypeOverride
    - SourceVersion
  - 回應語法
- StopBuildBatch
  - 請求語法
  - 支援的參數：
    - Id
  - 回應語法
- RetryBuildBatch
  - 請求語法
  - 支援的參數：
    - Id
    - IdempotencyToken
    - RetryType
  - 回應語法
- DeleteBuildBatch
  - 請求語法
  - 支援的參數：
    - Id
  - 回應語法

> **Note**
>
> 您可以使用 JSONPath 遞歸下降適用於 BatchDeleteBuilds 的 (..) 運算子。這會傳回一個陣列，並可讓您將 Arn 欄位從 StartBuild 轉換為 Ids 複數參數，如下所示。
>
> ```
> "BatchDeleteBuilds": {
> ```

```
    "Type": "Task",
    "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
    "Parameters": {
        "Ids.$": "$.Build..Arn"
    },
    "Next": "MyNextState"
},
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 呼叫 DynamoDB API

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> **Note**
>
> 「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

> **最佳化的 DynamoDB 整合與 SDK 整合有何不同 AWS**
>
> - 請求回應整合模式沒有最佳化。
> - 不支援等候傳回任務字符的回呼整合模式。
> - 只有`GetItem`、`PutItemUpdateItem`、和 `DeleteItem`API 動作可透過最佳化整合使用。其他 API 動作，例`CreateTable`如可透過 DynamoDB AWS SDK 整合取得。

支 Amazon DynamoDB 和語法：

- `GetItem`
  - 請求語法
  - 支援的參數：

    - Key
    - TableName
    - AttributesToGet
    - ConsistentRead
    - ExpressionAttributeNames
    - ProjectionExpression
    - ReturnConsumedCapacity
  - 回應語法
- PutItem
  - 請求語法
  - 支援的參數：
    - Item
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - 回應語法
- DeleteItem
  - 請求語法
  - 支援的參數：
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected

    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - 回應語法
- UpdateItem
  - 請求語法
  - 支援的參數：
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
    - UpdateExpression
  - 回應語法

> **Note**
>
> 中的參數Step Functions會在中表示 PascalCase，即使原生服務 API 位於 camelCase 中。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

以下是從 DynamoDB 擷取訊息的`Task`狀態。

```
    "Read Next Message from DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "TransferDataRecords-DDBTable-3I41R5L5EAGT",
        "Key": {
          "MessageId": {"S.$": "$.List[0]"}
        }
      },
      "ResultPath": "$.DynamoDB",
      "Next": "Send Message to SQS"
    },
```

若要查看運作範例中的這個狀態，請參閱 傳輸資料記錄 (Lambda、DynamoDB、Amazon SQS) 範例專案。

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 數管理 Amazon ECS 或 Fargate 任務

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

如何優化 Amazon ECS /法蓋特集成與Amazon ECS 或 Fargate SDK 集成不同 AWS

- 支援執行任務 (.sync)整合模式。
- `ecs:runTask`可以返回一個 HTTP 200 響應，但有一個非空`Failures`字段，如下所示：
  - 請求響應：返回響應，不會失敗任務。這與沒有優化相同。
  - 執行 Job 或工作 Token：如果遇到非空白`Failures`欄位，工作會失敗並顯示錯`AmazonECS.Unknown`誤。

支援的 Amazon EC/ Fargate API 和語法：

> **Note**
>
> 中的參數Step Functions會以表示 PascalCase，即使原生服務 API 位於 camelCase 中。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

- `RunTask` 會使用指定的任務定義來啟動新的任務。
  - 請求語法
  - 支援的參數：
    - `Cluster`
    - `Group`
    - `LaunchType`
    - `NetworkConfiguration`
    - `Overrides`
    - `PlacementConstraints`
    - `PlacementStrategy`
    - `PlatformVersion`
    - `PropagateTags`
    - `TaskDefinition`
    - `EnableExecuteCommand`
  - 回應語法

將資料傳遞至 Amazon ECS 任務

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

您可以使`overrides`用覆寫容器的預設命令，並將輸入傳遞至 Amazon ECS 任務。請參閱`ContainerOverride`。在這個例子中，我們已經用 JsonPath 來傳遞值到`Task`從輸入到`Task`狀態。

以下內容包括執行 Amazon ECS 任務並等待其完成的`Task`狀態。

```
{
```

```
 "StartAt": "Run an ECS Task and wait for it to complete",
 "States": {
   "Run an ECS Task and wait for it to complete": {
     "Type": "Task",
     "Resource": "arn:aws:states:::ecs:runTask.sync",
     "Parameters": {
                "Cluster": "cluster-arn",
                "TaskDefinition": "job-id",
                "Overrides": {
                    "ContainerOverrides": [
                        {
                            "Name": "container-name",
                            "Command.$": "$.commands"
                        }
                    ]
                }
            },
     "End": true
    }
  }
}
```

`ContainerOverrides` 中的 `"Command.$": "$.commands"` 行將命令從狀態輸入傳遞至容器。

針對先前的範例，如果執行的輸入如下所示，則每個命令將做為容器覆寫來傳遞：

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

以下內容包括執行 Amazon ECS 任務，然後等待任務權杖傳回的`Task`狀態。請參閱[等候傳回任務字符的回呼](#)。

```
{
   "StartAt":"Manage ECS task",
   "States":{
      "Manage ECS task":{
         "Type":"Task",
```

```
         "Resource":"arn:aws:states:::ecs:runTask.waitForTaskToken",
         "Parameters":{
            "LaunchType":"FARGATE",
            "Cluster":"cluster-arn",
            "TaskDefinition":"job-id",
            "Overrides":{
               "ContainerOverrides":[
                  {
                     "Name":"container-name",
                     "Environment":[
                        {
                           "Name":"TASK_TOKEN_ENV_VARIABLE",
                           "Value.$":"$$.Task.Token"
                        }
                     ]
                  }
               ]
            }
         },
         "End":true
      }
   }
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 呼叫 Amazon EMR

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> **最佳化的 Amazon EMR 整合與 Amazon EMR AWS 開發套件整合有何不同**
>
> 最佳化的 Amazon EMR 服務整合具有一組自訂的 API，用於包含基礎 Amazon EMR API，如下所述。因此，它與 Amazon EMR AWS 開發套件服務整合有很大不同。此外，也支援執行任務 (.sync)整合模式。

若要 AWS Step Functions 與 Amazon EMR 整合，您可以使用提供的 Amazon EMR 服務整合 API。服務整合 API 與對應的 Amazon EMR API 類似，傳遞的欄位和傳回的回應有些許差異。

如果停止執行，Step Functions 不會自動終止 Amazon EMR 叢集。如果您的狀態機器在 Amazon EMR 叢集終止之前停止，您的叢集可能會無限期地繼續執行，並可能產生額外費用。為避免這種情況，請確保您建立的任何 Amazon EMR 叢集都已正確終止。如需詳細資訊，請參閱：

- Amazon EMR 使用者指南中的控制叢集終止。
- 「服務整合模式執行任務 (.sync)」區段。

> **Note**
>
> 截至目前`emr-5.28.0`，您可以在建立叢集`StepConcurrencyLevel`時指定參數，以允許在單一叢集上 parallel 執行多個步驟。您可以使用 Step Functions `Map` 和`Parallel`狀態來提交與叢集 parallel 的工作。

Amazon EMR 服務整合的可用性取決於 Amazon EMR API 的可用性。如需特殊區域的限制，請參閱 Amazon EMR 文件。

> **Note**
>
> 為了與 Amazon EMR 整合，Step Functions 在此之後的前 10 分鐘和 300 秒內具有 60 秒的硬式編碼任務輪詢頻率。

下表說明每個服務整合 API 與其對應的 Amazon EMR API 之間的差異。

Amazon EMR 服務整合 API 和對應的 Amazon EMR API

Amazon EMR 服務整合 API	對應 EMR API	差異
createCluster 建立並開始執行叢集 (任務流程)。 Amazon EMR 會直接連結至稱為服務連結角色的唯一 IAM 角色類型。為了讓 `createCluster` 和 `createCluster.sync` 運	runJobFlow	`createCluster` 使用與下列項目相同的要求語法：runJobFlow • `Instances.KeepJobFlowAliveWhenNoSteps` 欄位是強制性的，且必須具有布林值 `TRUE`。 • 不允許 `Steps` 欄位。

Amazon EMR 服務整合 API	對應 EMR API	差異

作，您必須設定必要的許可來建立服務連結角色 AWSServiceRoleForEMRCleanup 。如需詳細資訊，包括可新增至 IAM 許可政策的陳述式，請參閱使用 Amazon EMR 的服務連結角色。		• 如果使用選用的 modifyInstanceFleetByName 連接器 API，則應提供 Instances.InstanceFleets[index].Name 欄位，且必須是唯一的。
		• 如果使用選用的 modifyInstanceGroupByName API，則應提供 Instances.InstanceGroups[index].Name 欄位，且必須是唯一的。
		
		回應為：
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR 使用這個：
		
		`{ "JobFlowId": "string" }`

createCluster.sync 建立並開始執行叢集 (任務流程)。	runJobFlow	與 createCluster 一樣，但會等待叢集到達 WAITING 狀態。

Amazon EMR 服務整合 API	對應 EMR API	差異

setClusterTermination保護
 	
鎖定叢集 (任務流程)，使叢集中的 EC2 執行個體無法藉由使用者介入、API 呼叫或任務流程錯誤而終止。
	setTerminationProtection	請求會使用：
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR 使用這個：
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster
 	
關閉叢集 (任務流程)。
	terminateJobFlows	請求會使用：
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR 使用這個：
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster.sync 關閉叢集 (任務流程)。	terminateJobFlows	與 `terminateCluster` 相同，但會等待叢集終止。

Amazon EMR 服務整合 API	對應 EMR API	差異

addStep
 	
新增步驟至執行中叢集。
 	
或者，您也可以在使用此 API 時指定Execution RoleArn 參數。
	addJobFlow步驟	請求使用密鑰"ClusterId" 。Amazon EMR 使用"JobFlowId" 。請求會使用單一步驟。
		
		`{ "Step": <"StepConfig object"> }`
		
		Amazon EMR 使用這個：
		
		`{ "Steps": [<StepConfig objects>] }`
		
		回應為：
		
		`{ "StepId": "string" }`
		
		Amazon EMR 返回這個：
		
		`{ "StepIds": [<strings>] }`

addStep.sync 新增步驟至執行中叢集。 或者，您也可以在使用此 API 時指定Execution RoleArn 參數。	addJobFlow步驟	與 addStep 相同，但會等待步驟完成。

Amazon EMR 服務整合 API	對應 EMR API	差異

cancelStep
 	
取消執行中叢集中的擱置步驟。
	cancelSteps	請求會使用：
		
		`{ "StepId": "string" }`
		
		Amazon EMR 使用這個：
		
		`{ "StepIds": [<strings>] }`
		
		回應為：
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		Amazon EMR 使用這個：
		
		`{ "CancelStepsInfoList": [<CancelStepsInfo objects>] }`

Amazon EMR 服務整合 API	對應 EMR API	差異
modifyInstanceFleetByName 針對具有所指定 `InstanceFleetName` 的執行個體機群，修改其目標隨需容量和目標 Spot 容量。	modifyInstanceFleet	請求與 `modifyInstanceFleet` 相同，除了： • 不允許 `Instance.InstanceFleetId` 欄位。 • 在執行時間，`InstanceFleetId` 是由服務整合自動決定，方法為呼叫 `ListInstanceFleets` 並剖析結果。

Amazon EMR 服務整合 API	對應 EMR API	差異

modifyInstanceGroupByName
 	
修改執行個體群組的節點數量和組態設定。
	modifyInstanceGroups	請求為：
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		Amazon EMR 使用一個列表：
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		在 InstanceGroupModifyConfig 物件中，不允許 InstanceGroupId 欄位。
		
		已新增新欄位 InstanceGroupName 。在執行時間，InstanceGroupId 是由服務整合自動決定，方法為呼叫 ListInstanceGroups 並剖析結果。

以下包含建立叢集的 Task 狀態。

```
"Create_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:createCluster.sync",
    "Parameters": {
```

```
        "Name": "MyWorkflowCluster",
        "VisibleToAllUsers": true,
        "ReleaseLabel": "emr-5.28.0",
        "Applications": [
            {
                "Name": "Hive"
            }
        ],
        "ServiceRole": "EMR_DefaultRole",
        "JobFlowRole": "EMR_EC2_DefaultRole",
        "LogUri": "s3n://aws-logs-123456789012-us-east-1/elasticmapreduce/",
        "Instances": {
            "KeepJobFlowAliveWhenNoSteps": true,
            "InstanceFleets": [
                {
                    "InstanceFleetType": "MASTER",
                    "Name": "MASTER",
                    "TargetOnDemandCapacity": 1,
                    "InstanceTypeConfigs": [
                        {
                            "InstanceType": "m4.xlarge"
                        }
                    ]
                },
                {
                    "InstanceFleetType": "CORE",
                    "Name": "CORE",
                    "TargetOnDemandCapacity": 1,
                    "InstanceTypeConfigs": [
                        {
                            "InstanceType": "m4.xlarge"
                        }
                    ]
                }
            ]
        }
    },
    "End": true
}
```

以下包含啟用終止保護的 Task 狀態。

```
"Enable_Termination_Protection": {
```

```
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

以下包含提交步驟至叢集的 Task 狀態。

```
"Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/myEMR-execution-role",
        "Step": {
            "Name": "The first step",
            "ActionOnFailure": "CONTINUE",
            "HadoopJarStep": {
                "Jar": "command-runner.jar",
                "Args": [
                    "hive-script",
                    "--run-hive-script",
                    "--args",
                    "-f",
                    "s3://<region>.elasticmapreduce.samples/cloudfront/code/
Hive_CloudFront.q",
                    "-d",
                    "INPUT=s3://<region>.elasticmapreduce.samples",
                    "-d",
                    "OUTPUT=s3://<mybucket>/MyHiveQueryResults/"
                ]
            }
        }
    },
    "End": true
}
```

以下包含取消步驟的 Task 狀態。

```
"Cancel_Step_One": {
```

```
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

以下包含終止叢集的 Task 狀態。

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

以下包含一種可對執行個體群組的叢集進行擴增或縮減的 Task 狀態。

```

"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

以下包含一種可對執行個體機群的叢集進行擴增或縮減的 Task 狀態。

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
```

```
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

在 EKS 上致電 Amazon EMR AWS Step Functions

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> EKS 整合上最佳化的 Amazon EMR 與 EKS SDK 整合上的 Amazon EMR 有何不同 AWS
>
> - 支援執行任務 (.sync)整合模式。
> - 請求回應整合模式沒有最佳化。
> - 不支援等候傳回任務字符的回呼整合模式。

> **Note**
>
> 為了與 Amazon EMR 整合，Step Functions 在此之後的前 10 分鐘和 300 秒內具有 60 秒的硬式編碼任務輪詢頻率。

若要 AWS Step Functions 與 EKS 上的 Amazon EMR 整合，請使用 Amazon EMR 上 EKS 服務整合 API。服務整合 API 與 EKS API 上對應的 Amazon EMR 相同，但並非所有 API 都支援所有整合模式，如下表所示。

API	請求回應	執行工作 (.sync)
CreateVirtualCluster	✓	

API	請求回應	執行工作 (.sync)
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

在 EKS API 上支持的 Amazon EMR：

> **Note**
>
> 「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

- CreateVirtualCluster
  - 請求語法
  - 支援的參數
  - 回應語法
- DeleteVirtualCluster
  - 請求語法
  - 支援的參數
  - 回應語法
- StartJobRun
  - 請求語法
  - 支援的參數
  - 回應語法

以下內容包括建Task立虛擬叢集的狀態。

```
"Create_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:createVirtualCluster",
  "Parameters": {
```

```
    "Name": "MyVirtualCluster",
    "ContainerProvider": {
      "Id": "EKSClusterName",
      "Type": "EKS",
      "Info": {
        "EksInfo": {
          "Namespace": "Namespace"
        }
      }
    }
  },
  "End": true
}
```

以下內容包括將工作送出至虛擬叢集並等待其完成的Task狀態。

```
"Submit_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-containers:startJobRun.sync",
    "Parameters": {
      "Name": "MyJobName",
      "VirtualClusterId.$": "$.VirtualClusterId",
      "ExecutionRoleArn": "arn:aws:iam::<accountId>:role/job-execution-role",
      "ReleaseLabel": "emr-6.2.0-latest",
      "JobDriver": {
        "SparkSubmitJobDriver": {
          "EntryPoint": "s3://<mybucket>/jobs/trip-count.py",
          "EntryPointArguments": [
            "60"
          ],
          "SparkSubmitParameters": "--conf spark.driver.cores=2 --conf
 spark.executor.instances=10 --conf spark.kubernetes.pyspark.pythonVersion=3 --conf
 spark.executor.memory=10G --conf spark.driver.memory=10G --conf spark.executor.cores=1
 --conf spark.dynamicAllocation.enabled=false"
        }
      },
      "ConfigurationOverrides": {
        "ApplicationConfiguration": [
          {
            "Classification": "spark-defaults",
            "Properties": {
              "spark.executor.instances": "2",
              "spark.executor.memory": "2G"
```

```
            }
          }
        ],
        "MonitoringConfiguration": {
          "PersistentAppUI": "ENABLED",
          "CloudWatchMonitoringConfiguration": {
            "LogGroupName": "MyLogGroupName",
            "LogStreamNamePrefix": "MyLogStreamNamePrefix"
          },
          "S3MonitoringConfiguration": {
            "LogUri": "s3://<mylogsbucket>"
          }
        }
      },
      "Tags": {
        "taskType": "jobName"
      }
    },
    "End": true
}
```

以下內容包括刪除虛擬叢集並等待刪除完成的Task狀態。

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 調用 Amazon EKS

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

優化的 Amazon EKS 整合與 Amazon EKS 開發套件整合有何不同 AWS

- 支援執行任務 (.sync)整合模式。
- 請求回應整合模式沒有最佳化。
- 不支援等候傳回任務字符的回呼整合模式。

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

Step Functions 提供兩種類型的服務整合 API，可與 Amazon Elastic Kubernetes Service 整合。一個允許您使用 Amazon EKS API 來創建和管理 Amazon EKS 集群。另一個可讓您使用 Kubernetes API 與叢集互動，並在應用程式工作流程中執行作業。您可以將 Kubernetes API 整合與使用 Step Functions 建立的 Amazon EKS 叢集搭配使用 eksctl 工具或 Amazon EKS 主控台建立的 Amazon EKS 叢集，或類似的方法。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的建立 Amazon EKS 叢集。

Note

Step Functions EKS 整合僅支援具有公用端點存取權的 Kubernetes API。依預設，EKS 叢集 API 伺服器端點具有公開存取權。如需詳細資訊，請參閱 Amazon EKS 叢集端點存取控制 (英文) 中的 Amazon EKS 叢集端點存取控制。

如果停止執行，Step Functions 不會自動終止 Amazon EKS 叢集。如果您的狀態機器在 Amazon EKS 叢集終止之前停止，您的叢集可能會無限期地繼續執行，並可能產生額外費用。為避免這種情況，請確保您建立的任何 Amazon EKS 叢集都已正確終止。如需詳細資訊，請參閱：

- 刪除 Amazon EKS 使用者指南中的叢集。
- 執行任務 (.sync)在服務集成模式。

Note

「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

庫伯內特斯 API 整合

Step Functions 支援下列庫伯內特 API：

RunJob

`eks:runJob`服務整合可讓您在 Amazon EKS 叢集上執行任務。該`eks:runJob.sync`變體允許您等待工作完成，並選擇性地檢索日誌。

您的 Kubernetes API 伺服器必須將許可授與狀態機器使用的 IAM 角色。如需詳細資訊，請參閱 許可。

對於執行 Job (`.sync`) 模式，工作的狀態是由輪詢決定。Step Functions 最初以每分鐘大約 1 次輪詢的速率進行輪詢。這個速率最終會減慢到大約每 5 分鐘進行 1 次調查。如果您需要更頻繁的輪詢，或需要對輪詢策略進行更多控制，則可以使用`eks:call`整合來查詢工作的狀態。

此`eks:runJob`整合特定於`batch/v1`庫伯尼特工作。如需詳細資訊，請參閱 Kubernetes 文件中的工作。如果您想要管理其他 Kubernetes 資源 (包括自訂資源)，請使用服務整合`eks:call`。您可以使用 Step Functions 來建立輪詢迴圈，如the section called “投票 Job 狀態 (Lambda、 AWS Batch) ”範例專案中所示。

支援的參數包括：

- `ClusterName`：您要呼叫的 Amazon EKS 叢集的名稱。
  - `Type: String`
  - 必要：是
- `CertificateAuthority`：與叢集通訊所需的 Base64 編碼憑證資料。您可以從 Amazon EKS 控制台或使用 Amazon EKS API 獲取此值。DescribeCluster
  - `Type: String`
  - 必要：是
- `Endpoint`：您的 Kubernetes API 伺服器的端點網址。您可以從 Amazon EKS 控制台或使用 Amazon EKS API 獲取此值。DescribeCluster
  - `Type: String`
  - 必要：是
- `Namespace`：執行工作的命名空間。如果未提供，則會使`default`用命名空間。
  - `Type: String`
  - 必要：否

- Job：Job 的定義。請參閱 Kubernetes 文件中的工作。
  - Type：JSON 或 String
  - 必要：是
- LogOptions：用於控制日誌的可選檢索的一組選項。只有在使用「執行 Job (.sync)」服務整合模式來等待工作完成時才適用。
  - Type: JSON
  - 必要：否
  - 記錄會包含在金鑰下的回應中logs。工作中可能有多個網繭，每個 Pod 都有多個容器。

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
        }
      },
      ...
    }
  }
```

  - 記錄擷取是以最大的方式執行。如果擷取記錄檔時發生錯誤，則會有log欄位error和cause。
- LogOptions.RetrieveLogs：在工作完成後啟用記錄擷取。依預設，不會擷取記錄檔。
  - Type: Boolean
  - 必要：否
- LogOptions.RawLogs：如果設定RawLogs為 true，則會以原始字串傳回記錄，而不會嘗試將記錄剖析為 JSON。根據預設，記錄會在可能的情況下還原序列化為 JSON。在某些情況下，這種解析可能會引入不必要的更改，例如限制包含多個數字的數字的精度。
  - Type: Boolean
  - 必要：否
- LogOptions.LogParameters：Kubernetes API 的讀取記錄 API 支援查詢參數以控制記錄擷取。例如，您可以使用tailLines或limitBytes限制擷取記錄檔的大小，並保留在 Step Functions 資料大小配額內。如需詳細資訊，請參閱 Kubernetes API 參考資料的「讀取記錄」一節。

- Type: String到的地圖 List of Strings
- 必要：否
- 範例：

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

下列範例包含執行工作、等待工作完成，然後擷取工作記錄的Task狀態：

```
{
  "StartAt": "Run a job on EKS",
  "States": {
    "Run a job on EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:runJob.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://AKIAIOSFODNN7EXAMPLE.yl4.us-east-1.eks.amazonaws.com",
        "LogOptions": {
          "RetrieveLogs": true
        },
        "Job": {
          "apiVersion": "batch/v1",
          "kind": "Job",
          "metadata": {
            "name": "example-job"
          },
          "spec": {
            "backoffLimit": 0,
            "template": {
              "metadata": {
                "name": "example-job"
              },
              "spec": {
                "containers": [
                  {
                    "name": "pi-2000",
                    "image": "perl",
                    "command": [ "perl" ],
```

```
                    "args": [
                      "-Mbignum=bpi",
                      "-wle",
                      "print bpi(2000)"
                    ]
                  }
                ],
                "restartPolicy": "Never"
              }
            }
          }
        }
      },
      "End": true
    }
  }
}
```

Call

eks:call服務整合可讓您使用 Kubernetes API 透過 Kubernetes API 端點讀取和寫入 Kubernetes 資源物件。

您的 Kubernetes API 伺服器必須將許可授與狀態機器使用的 IAM 角色。如需詳細資訊，請參閱 許可。

如需有關可用作業的詳細資訊，請參閱 Kubernetes API 參考資料。

支援的參數Call包括：

- ClusterName：您要呼叫的 Amazon EKS 叢集的名稱。
  - Type：String
  - 必要：是
- CertificateAuthority：與叢集通訊所需的 Base64 編碼憑證資料。您可以從 Amazon EKS 控制台或使用 Amazon EKS API 獲取此值。DescribeCluster
  - Type: String
  - 必要：是
- Endpoint：您的 Kubernetes API 伺服器的端點網址。您可以在 Amazon EKS 控制台或使用 Amazon EKS 的 API 找到此值。 DescribeCluster
  - Type: String

  - 必要：是
- Method：您的請求的 HTTP 方法。下列其中一項：GET、POST、PUT、DELETE、HEAD 或 PATCH。
  - Type: String
  - 必要：是
- Path：執行 REST API 作業的 HTTP 路徑。
  - Type: String
  - 必要：是
- QueryParameters：執行 REST API 作業的 HTTP 查詢參數。
  - Type: String到的地圖 List of Strings
  - 必要：否
  - 範例：

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- RequestBody：執行 REST API 作業的 HTTP 訊息內文。
  - Type：JSON 或 String
  - 必要：否

以下包含用eks:call來列出屬於工作之網繭的Task狀態example-job。

```
{
  "StartAt": "Call EKS",
  "States": {
    "Call EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "GET",
        "Path": "/api/v1/namespaces/default/pods",
        "QueryParameters": {
          "labelSelector": [
```

```
            "job-name=example-job"
          ]
        }
      },
      "End": true
    }
  }
}
```

以下內容包括用eks:call來刪除工作的Task狀態example-job，並設定以確propagationPolicy保工作的網繭也會遭到刪除。

```
{
  "StartAt": "Call EKS",
  "States": {
    "Call EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "DELETE",
        "Path": "/apis/batch/v1/namespaces/default/jobs/example-job",
        "QueryParameters": {
          "propagationPolicy": [
            "Foreground"
          ]
        }
      },
      "End": true
    }
  }
}
```

支援的 Amazon EKS API

支援的 Amazon EKS API 和語法包括：

- CreateCluster
  - 請求語法
  - 回應語法

使用eks:createCluster服務整合建立 Amazon EKS 叢集時，IAM 角色會以管理員身分新增至 Kubernetes RBAC 授權表格 (具有系統:主要許可)。一開始，只有該 IAM 實體可以呼叫 Kubernetes API 伺服器。如需詳細資訊，請參閱：

- 在 Amazon EKS 使用者指南中管理叢集的使用者或 IAM 角色
- 該許可部分

Amazon EKS 使用服務連結角色，其中包含 Amazon EKS 代表您呼叫其他服務所需的許可。如果這些服務連結角色不存在於您的帳戶中，您必須將iam:CreateServiceLinkedRole權限新增至 Step Functions 使用的 IAM 角色。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的使用服務連結角色。

Step Functions 使用的 IAM 角色必須具有iam:PassRole許可，才能將叢集 IAM 角色傳遞給 Amazon EKS。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的 Amazon EKS 叢集 IAM 角色。

- DeleteCluster
  - 請求語法
  - 回應語法

  您必須先刪除任何 Fargate 設定檔或節點群組，才能刪除叢集。
- CreateFargateProfile
  - 請求語法
  - 回應語法

  Amazon EKS 使用服務連結角色，其中包含 Amazon EKS 代表您呼叫其他服務所需的許可。如果這些服務連結角色不存在於您的帳戶中，您必須將iam:CreateServiceLinkedRole權限新增至 Step Functions 使用的 IAM 角色。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的使用服務連結角色。

  Fargate 上的 Amazon EKS 可能無法在所有地區提供。如需區域可用性的相關資訊，請參閱 Amazon EKS 使用者指南中關於 Fargate 的章節。

  Step Functions 使用的 IAM 角色必須具有iam:PassRole許可，才能將網繭執行 IAM 角色傳遞給 Amazon EKS。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的網繭執行角色。
- DeleteFargateProfile
  - 請求語法
  - 回應語法

- CreateNodegroup
  - 請求語法
  - 回應語法

  Amazon EKS 使用服務連結角色，其中包含 Amazon EKS 代表您呼叫其他服務所需的許可。如果這些服務連結角色不存在於您的帳戶中，您必須將`iam:CreateServiceLinkedRole`權限新增至 Step Functions 使用的 IAM 角色。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的使用服務連結角色。

  Step Functions 使用的 IAM 角色必須具有`iam:PassRole`許可，才能將節點 IAM 角色傳遞給 Amazon EKS。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的使用服務連結角色。
- DeleteNodegroup
  - 請求語法
  - 回應語法

以下內容包括`Task`可建立 Amazon EKS 叢集的項目。

```
{
  "StartAt": "CreateCluster.sync",
  "States": {
    "CreateCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createCluster.sync",
      "Parameters": {
        "Name": "MyCluster",
        "ResourcesVpcConfig": {
          "SubnetIds": [
            "subnet-053e7c47012341234",
            "subnet-027cfea4b12341234"
          ]
        },
        "RoleArn": "arn:aws:iam::123456789012:role/MyEKSClusterRole"
      },
      "End": true
    }
  }
}
```

以下內容包括刪除 Amazon EKS 叢集的`Task`狀態。

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
      },
      "End": true
    }
  }
}
```

以下內容包括建Task立 Fargate 設定檔的狀態。

```
{
  "StartAt": "CreateFargateProfile.sync",
  "States": {
    "CreateFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile",
        "PodExecutionRoleArn": "arn:aws:iam::123456789012:role/
MyFargatePodExecutionRole",
        "Selectors": [{
            "Namespace": "my-namespace",
            "Labels": { "my-label": "my-value" }
          }]
      },
      "End": true
    }
  }
}
```

以下內容包括刪除 Fargate 設定檔的Task狀態。

```
{
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
```

```
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
  }
}
```

以下內容包括建Task立節點群組的狀態。

```
{
  "StartAt": "CreateNodegroup.sync",
  "States": {
    "CreateNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup",
        "NodeRole": "arn:aws:iam::123456789012:role/MyNodeInstanceRole",
        "Subnets": ["subnet-09fb51df01234", "subnet-027cfea4b1234"]
      },
      "End": true
    }
  }
}
```

以下內容包括刪除節點群組的Task狀態。

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup"
      },
```

```
      "End": true
    }
  }
}
```

許可

使用`eks:createCluster`服務整合建立 Amazon EKS 叢集時，IAM 角色會以管理員身分新增至 Kubernetes RBAC 授權資料表，並具有許可。`system:masters`一開始，只有該 IAM 實體可以呼叫 Kubernetes API 伺服器。例如，您將無法使用 kubectl 與 Ku bernetes API 伺服器互動，除非您承擔與 Step Functions 狀態機器相同的角色，或者設定 Kubernetes 將許可授與其他 IAM 實體。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的管理叢集的使用者或 IAM 角色。

您可以將其他 IAM 實體 (例如使用者或角色) 新增權限，方法是將這些實體新增至 kube-system 命名空`aws-authConfigMap`間中的。如果您要從 Step Functions 建立叢集，請使用`eks:call`服務整合。

以下包含建`Task`立`aws-authConfigMap`並授與使用者`arn:aws:iam::123456789012:user/my-user`和 IAM 角色`system:masters`權限的狀態`arn:aws:iam::123456789012:role/my-role`。

```
{
  "StartAt": "Add authorized user",
  "States": {
    "Add authorized user": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "LS0tLS1CRUd...UtLS0tLQo=",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "POST",
        "Path": "/api/v1/namespaces/kube-system/configmaps",
        "RequestBody": {
           "apiVersion": "v1",
           "kind": "ConfigMap",
           "metadata": {
              "name": "aws-auth",
              "namespace": "kube-system"
           },
           "data": {
             "mapUsers": "[{ \"userarn\": \"arn:aws:iam::123456789012:user/my-user\",
 \"username\": \"my-user\", \"groups\": [ \"system:masters\" ] } ]",
```

```
            "mapRoles": "[{ \"rolearn\": \"arn:aws:iam::123456789012:role/my-role\",
\"username\": \"my-role\", \"groups\": [ \"system:masters\" ] } ]"
          }
        }
      },
      "End": true
    }
  }
```

> **Note**
>
> 您可能會看到 IAM 角色的 ARN 以包含路徑 /服務角色/ 的格式顯示，例如。`arn:aws:iam::123456789012:role/service-role/my-role`在`aws-auth`中列出角色時，不應包含此服務角色路徑 Token。

第一次建立叢集時，`aws-auth`ConfigMap將不存在，但如果您建立 Fargate 設定檔，則會自動新增叢集。您可以擷取的目前值`aws-auth`、新增其他權限以及PUT新版本。通常在 Fargate 配置文件`aws-auth`之前創建更容易。

如果您的叢集是在 Step Functions 之外建立的，您可以設定 kubectl 與您的 Kubernetes API 伺服器進行通訊。然後，使用創建一個新的`kubectl apply -f aws-auth.yaml`或`aws-auth`ConfigMap使用編輯已存在的`kubectl edit -n kube-system configmap/aws-auth`。如需詳細資訊，請參閱：

- 在 Amazon EKS 用戶指南中為 Amazon EKS 創建庫貝配置。
- 在 Amazon EKS 使用者指南中管理叢集的使用者或 IAM 角色。

如果您的 IAM 角色在 Kubernetes 中沒有足夠的許可，`eks:call`或`eks:runJob`服務整合將失敗，並出現以下錯誤：

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
    "kind": "Status",
    "apiVersion": "v1",
```

```
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
    "code": 401
  },
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

使用 Step Functions 呼叫 Amazon EMR Serverless

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> 最佳化EMR Serverless整合與 EMR Serverless AWS SDK 整合有何不同
>
> - 最佳化EMR Serverless服務整合具有一組自訂的 API，用於包裝基礎 EMR Serverless API。由於此自訂，最佳化的EMR Serverless整合與 EMR Serverless AWS SDK 服務整合有很大差異。此外，優化的EMR Serverless集成支持執行任務 (.sync)集成模式。
> - 不支援等候傳回任務字符的回呼整合模式。

在本主題中

EMR Serverless服務整合 API

若要AWS Step Functions與整合EMR Serverless，您可以使用下列六個EMR Serverless服務整合 API。這些服務整合 API 類似於對應的 EMR Serverless API，在傳遞的欄位和傳回的回應中有些許差異。

下表描述每個服務整合 API 與其對應 EMR Serverless API 之間的差異：

EMR Serverless服務整合 API 和對應的 EMR Serverless API

EMR Serverless服務整合 API	對應的 EMR Serverless API	差異
創建應用 建立應用程式。 EMR Serverless連結至稱為服務連結IAM角色的唯一角色類型。為了讓 `createApplication` 和 `createApplication.sync` 運作，您必須設定必要的許可來建立服務連結角色 `AWS ServiceRoleForAmazonEMRServerless` 。如需有關此項目的詳細資訊，包括您可以新增至IAM權限原則的陳述式，請參閱使用的服務連結角色。EMR Serverless	CreateApplication	無
建立應用程式. 同步 建立應用程式。	CreateApplication	API 和EMR Serverless服務整合 EMR Serverless API 的請求和回應之間沒有差異。不過，建立應用程式 .sync 會等待應用程式到達狀態。`CREATED`
開始應用 啟動指定的應用程式，並初始化應用程式的初始容量 (若已設定)。	StartApplication	EMR ServerlessAPI 回應不包含任何資料，但EMR Serverless服務整合 API 回應包含下列資料。 `{ "ApplicationId": "string" }`

EMR Serverless服務整合 API	對應的 EMR Serverless API	差異
開始應用程式. 同步 啟動指定的應用程式並初始化初始容量 (如果已設定)。	StartApplication	EMR ServerlessAPI 回應不包含任何資料，但EMR Serverless服務整合 API 回應包含下列資料。 `{ "ApplicationId": "string" }` 此外，啟動應用程式 .sync 會等待應用程式到達狀態。STARTED
停止申請 停止指定的應用程式並釋放初始容量 (若已設定)。停止應用程式之前，必須完成或取消所有已排定和執行中的工作。	StopApplication	EMR ServerlessAPI 回應不包含任何資料，但EMR Serverless服務整合 API 回應包含下列資料。 `{ "ApplicationId": "string" }`

EMR Serverless服務整合 API	對應的 EMR Serverless API	差異
停止應用程式. 同步 停止指定的應用程式並釋放初始容量 (若已設定)。停止應用程式之前，必須完成或取消所有已排定和執行中的工作。	StopApplication	EMR ServerlessAPI 回應不包含任何資料，但EMR Serverless服務整合 API 回應包含下列資料。 `{ "ApplicationId": "string" }` 此外，停止應用程序 .sync 等待應用程序到達狀態。STOPPED
刪除應用程式 刪除應用程式。應用程式必須處於STOPPED或CREATED狀態，才能刪除。	DeleteApplication	EMR ServerlessAPI 回應不包含任何資料，但EMR Serverless服務整合 API 回應包含下列資料。 `{ "ApplicationId": "string" }`

EMR Serverless服務整合 API	對應的 EMR Serverless API	差異
刪除應用程式. 同步 刪除應用程式。應用程式必須處於STOPPED或CREATED狀態，才能刪除。	DeleteApplication	EMR ServerlessAPI 回應不包含任何資料，但EMR Serverless服務整合 API 回應包含下列資料。 `{ "ApplicationId": "string" }` 此外，停止應用程序 .sync 等待應用程序到達狀態。TERMINATED
startJobRun 開始工作執行。	StartJobRun	無
startJobRun. 同步 開始工作執行。	StartJobRun	API 和EMR Serverless服務整合 EMR Serverless API 的請求和回應之間沒有差異。不過，startJobRun.sync 會等待應用程式到達狀態SUCCESS。
cancelJobRun 取消工作執行。	CancelJobRun	無
cancelJobRun. 同步 取消工作執行。	CancelJobRun	API 和EMR Serverless服務整合 EMR Serverless API 的請求和回應之間沒有差異。不過，cancelJob Run.sync 會等待應用程式到達狀態CANCELLED 。

EMR 無伺服器整合使用案例

針對「最佳化」EMR Serverless 服務整合，建議您建立單一應用程式，然後使用該應用程式來執行多個工作。例如，在單一狀態機器中，您可以包含多個startJobRun要求，所有要求都使用相同的應用程式。下列任務狀態範例顯示了與 EMR Serverless API 整合的使用案例Step Functions。如需的其他使用案例的詳細資訊EMR Serverless，請參閱什麼是Amazon EMR Serverless。

> **Tip**
>
> 若要部署與執行多個作業整EMR Serverless合的狀態機器範例 AWS 帳戶，請參閱執行EMR Serverless工作。

- 建立應用程式
- 啟動應用程式
- 停止應用程式
- 刪除應用程式
- 在應用程式中啟動工作
- 取消應用程式中的工作

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

在下列使用案例中顯示的範例中，以您的資源特定資訊取代##文字。例如，用您*yourApplicationId*的EMR Serverless應用程序的 ID 替換，例如`00yv7iv71inak893`。

建立應用程式

下列工作狀態範例會使用建立應用程式 .sync 服務整合 API 建立應用程式。

```
"Create_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:createApplication.sync",
    "Parameters": {
        "Name": "MyApplication",
        "ReleaseLabel": "emr-6.9.0",
        "Type": "SPARK"
    },
```

```
    "End": true
}
```

啟動應用程式

下列工作狀態範例會使用啟動應用程式 .sync 服務整合 API 來啟動應用程式。

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

停止應用程式

下列工作狀態範例會停止使用停止應用程式 .sync 服務整合 API 的應用程式。

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

刪除應用程式

下列工作狀態範例會使用刪除應用程式 .sync 服務整合 API 刪除應用程式。

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

在應用程式中啟動工作

下列工作狀態範例會使用 startJobRun.sync 服務整合 API 在應用程式中啟動工作。

```
"Start_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startJobRun.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId",
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/myEMRServerless-execution-
role",
        "JobDriver": {
            "SparkSubmit": {
                "EntryPoint": "s3://<mybucket>/sample.py",
                "EntryPointArguments": ["1"],
                "SparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=4g --conf spark.driver.cores=2 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
            }
        }
    },
    "End": true
}
```

取消應用程式中的工作

下列工作狀態範例會使用 cancelJobRun.sync 服務整合 API 取消應用程式中的工作。

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

EventBridge 使用 Step Functions 調用

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

最佳化 EventBridge 整合與 EventBridge AWS SDK 整合有何不同

- 執行 ARN 和狀態機 ARN 會自動附加到每個字`Resources`段中。`PutEventsRequestEntry`
- 如果來自的響應`PutEvents`包含非零，`FailedEntryCount`則`Task`狀態失敗並顯示錯誤`EventBridge.FailedEntry`。

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

Step Functions 提供與 Amazon 整合的服務整合 API EventBridge。這可讓您直接從 Step Functions 工作流程傳送自訂事件，以建置事件導向的應用程式。

若要使用 `PutEvents` API，您必須在帳戶中建立符合要傳送之事件特定模式的 EventBridge 規則。例如，您可以：

- 在您的帳戶中建立 Lambda 函數，以接收和列印符合 EventBridge 規則的事件。
- 在您的帳戶中建立符合特定事件模式的預設事件匯流排上的 EventBridge 規則，並以 Lambda 函數為目標。

如需詳細資訊，請參閱：

- 在 EventBridge 用戶指南 PutEvents中添加 Amazon EventBridge 事件。
- 等候傳回任務字符的回呼在服務集成模式。

Note

「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

支援的 EventBridge API

支援的 EventBridge API 和語法包括：

- PutEvents
  - 請求語法
  - 支持的參數：
    - Entries
  - 回應語法

以下內容包括傳送自訂事件的項目：Task

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

錯誤處理

PutEventsAPI 接受一組項目作為輸入，然後返回結果條目的數組。只要PutEvents動作成功，就PutEvents會傳回 HTTP 200 回應，即使一個或多個項目失敗。 PutEvents傳回欄位中失敗項目的FailedEntryCount數目。

Step Functions 檢查FailedEntryCount是否大於零。如果大於零，則 Step Functions 會使狀態失敗，並顯示錯誤EventBridge.FailedEntry。這可讓您在工作狀態上使用 Step Functions 的內建錯誤處理，在發生失敗的項目時 catch 取或重試，而不需要使用其他狀態FailedEntryCount來分析回應。

> **Note**
>
> 如果您已經實現了冪等性並且可以安全地重試所有條目，則可以使用步驟函數的重試邏輯。Step Functions 在重試之前不會從`PutEvents`輸入陣列中移除成功的條目。相反地，它會使用原始的項目陣列重試。

使用 Step Functions 管理 AWS Glue 工作

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> **最佳化 AWS Glue 整合與 AWS GlueAWS SDK 整合有何不同**
>
> - 整執行任務 (.sync)合模式可用。
> - 該`JobName`字段從請求中提取並插入到響應中，該響應通常只包含`JobRunID`。

支持的 AWS Glue API：

- `StartJobRun`

> **Note**
>
> 中的參數Step Functions會在中表示 PascalCase，即使原生服務 API 位於 camelCase 中。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

以下包含啟動 AWS Glue 工作的`Task`狀態。

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
```

```
      "Next": "ValidateOutput"
    },
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 管理 AWS Glue DataBrew 工作

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

您可以使用 DataBrew 整合將資料清理和資料標準化步驟新增至您的分析和機器學習工作流程。

支持的 DataBrew API：

- `StartJobRun`

以下內容包含啟動要求-回應 DataBrew工作的`Task`狀態。

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
            "Parameters": {
               "Name": "sample-proj-job-1"
            },
            "Next": "NEXT_STATE"
          },
```

以下包含啟動同步 DataBrew 工作的`Task`狀態。

```
"DataBrew StartJobRun": {
           "Type": "Task",
           "Resource": "arn:aws:states:::databrew:startJobRun.sync",
           "Parameters": {
              "Name": "sample-proj-job-1"
           },
           "Next": "NEXT_STATE"
          },
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 叫用 Lambda

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> 最佳化 Lambda 整合與 Lambda AWS 開發套件整合有何不同
>
> - 響應Payload字段從轉義的 Json 解析為 Json。
> - 如果回應包含欄位，FunctionError或在 Lambda 函數中引發例外狀況，則工作會失敗。

如需管理狀態輸入、輸出和結果的詳細資訊，請參閱Step Functions 中的輸入和輸出處理。

支援的 AWS Lambda API：

- Invoke
  - 請求語法
  - 支援的參數
    - ClientContext
    - FunctionName
    - InvocationType
    - Qualifier
    - Payload
  - 回應語法

> Note
>
> 中的參數Step Functions會在中表示 PascalCase，即使原生服務 API 位於駝峰中也是如此。例如，您可以使用 Step Functions API 動作startSyncExecution並將其參數指定為StateMachineArn。

以下內容包含叫用 Lambda 函數的Task狀態。

```
{
   "StartAt":"CallLambda",
   "States":{
      "CallLambda":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-1:123456789012:function:MyFunction"
         },
         "End":true
      }
   }
}
```

以下包含 Task 狀態，此狀態會實作回呼服務整合模式。

```
{
   "StartAt":"GetManualReview",
   "States":{
      "GetManualReview":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke.waitForTaskToken",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-1:123456789012:function:get-model-
review-decision",
            "Payload":{
               "model.$":"$.new_model",
               "token.$":"$$.Task.Token"
            },
            "Qualifier":"prod-v1"
         },
         "End":true
      }
   }
}
```

當您叫用 Lambda 函數時，執行會等待函數完成。如果您使用回呼工作叫用 Lambda 函數，則在 Lambda 函數完成執行並傳回結果之前，活動訊號逾時才會開始計算。只要執行 Lambda 函數，就不會強制執行活動訊號逾時。

您也可以使用InvocationType參數以非同步方式呼叫 Lambda，如下列範例所示：

> **Note**
>
> 對於 Lambda 函數的非同步叫用，活動訊號逾時期間會立即開始。

```
{

  "Comment": "A Hello World example of the Amazon States Language using Pass states",
  "StartAt": "Hello",
  "States": {
    "Hello": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-1:123456789012:function:echo",
        "InvocationType": "Event"
      },
      "End": true
    }
  }
}
```

傳回Task結果時，函數輸出會嵌套在中繼資料字典中。例如：

```
{

   "ExecutedVersion":"$LATEST",
   "Payload":"FUNCTION OUTPUT",
   "SdkHttpMetadata":{
      "HttpHeaders":{
         "Connection":"keep-alive",
         "Content-Length":"4",
         "Content-Type":"application/json",
         "Date":"Fri, 26 Mar 2021 07:42:02 GMT",
         "X-Amz-Executed-Version":"$LATEST",
         "x-amzn-Remapped-Content-Length":"0",
         "x-amzn-RequestId":"0101aa0101-1111-111a-aa55-1010aaa1010",
         "X-Amzn-Trace-Id":"root=1-1a1a000a2a2-fe0101aa10ab;sampled=0"
      },
      "HttpStatusCode":200
   },
   "SdkResponseMetadata":{
```

```
      "RequestId":"6b3bebdb-9251-453a-ae45-512d9e2bf4d3"
   },
   "StatusCode":200
}
```

或者，您可以直接在「資源」欄位中指定函數 ARN 來叫用 Lambda 函數。當您以這種方式叫用 Lambda 函數時，您無法指定.waitForTaskToken，且工作結果只包含函數輸出。

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

您可以在Resource欄位中的 ARN 中指定這些選項，以叫用特定的 Lambda 函數版本或別名。請參閱 Lambda 文件中的下列內容：

- AWS Lambda 版本控制
- AWS Lambda 別名

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

SageMaker 使用 Step Functions 管理

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> 最佳化 SageMaker 整合與 SageMaker AWS SDK 整合有何不同
>
> - 支援執行任務 (.sync)整合模式。
> - 請求回應整合模式沒有最佳化。
> - 不支援等候傳回任務字符的回呼整合模式。

支援的 SageMaker API 和語法：

- CreateEndpoint
  - 請求語法
  - 支援的參數：
    - EndpointConfigName
    - EndpointName
    - Tags
  - 回應語法
- CreateEndpointConfig
  - 請求語法
  - 支援的參數：
    - EndpointConfigName
    - KmsKeyId
    - ProductionVariants
    - Tags
  - 回應語法
- CreateHyperParameterTuningJob

> **Note**
>
> 此 API 動作支援整.sync合模式。

  - 請求語法
  - 支援的參數：
    - HyperParameterTuningJobConfig
    - HyperParameterTuningJobName
    - Tags
    - TrainingJobDefinition
    - WarmStartConfig
  - 回應語法

- CreateLabelingJob

  > **Note**
  >
  > 此 API 動作支援整.sync合模式。

  - 請求語法
  - 支援的參數：
    - HumanTaskConfig
    - InputConfig
    - LabelAttributeName
    - LabelCategoryConfigS3Uri
    - LabelingJobAlgorithmsConfig
    - LabelingJobName
    - OutputConfig
    - RoleArn
    - StoppingConditions
    - Tags
  - 回應語法
- CreateModel
  - 請求語法
  - 支援的參數：
    - Containers
    - EnableNetworkIsolation
    - ExecutionRoleArn
    - ModelName
    - PrimaryContainer
    - Tags
    - VpcConfig
- CreateProcessingJob

> **Note**
>
> 此 API 動作支援整.sync合模式。

- 請求語法
- 支援的參數：
  - AppSpecification
  - Environment
  - ExperimentConfig
  - NetworkConfig
  - ProcessingInputs
  - ProcessingJobName
  - ProcessingOutputConfig
  - ProcessingResources
  - RoleArn
  - StoppingCondition
  - Tags
- 回應語法

- CreateTrainingJob

> **Note**
>
> 此 API 動作支援整.sync合模式。

- 請求語法
- 支援的參數：
  - AlgorithmSpecification
  - HyperParameters
  - InputDataConfig
  - OutputDataConfig

    - ResourceConfig
    - RoleArn
    - StoppingCondition
    - Tags
    - TrainingJobName
    - VpcConfig
  - 回應語法
- CreateTransformJob

> **Note**
>
> 此 API 動作支援整.sync合模式。

> **Note**
>
> AWS Step Functions 不會自動建立的策略CreateTransformJob。您必須將內嵌政策附加到建立的角色。如需詳細資訊，請參閱此 IAM 政策範例：CreateTrainingJob。

  - 請求語法
  - 支援的參數：
    - BatchStrategy
    - Environment
    - MaxConcurrentTransforms
    - MaxPayloadInMB
    - ModelName
    - Tags
    - TransformInput
    - TransformJobName
    - TransformOutput
    - TransformResources
  - 回應語法

- UpdateEndpoint
  - 請求語法
  - 支援的參數：
    - EndpointConfigName
    - EndpointName
  - 回應語法

SageMaker 轉換 Job 範例

以下內容包括建立 Amazon SageMaker 轉換任務的Task狀態，並指定DataSource和的 Amazon S3 位置TransformOutput。

```
{
"SageMaker CreateTransformJob": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sagemaker:createTransformJob.sync",
  "Parameters": {
    "ModelName": "SageMakerCreateTransformJobModel-9iFBKsYti9vr",
    "TransformInput": {
      "CompressionType": "None",
      "ContentType": "text/csv",
      "DataSource": {
        "S3DataSource": {
          "S3DataType": "S3Prefix",
          "S3Uri": "s3://my-s3bucket-example-1/TransformJobDataInput.txt"
        }
      }
    },
    "TransformOutput": {
      "S3OutputPath": "s3://my-s3bucket-example-1/TransformJobOutputPath"
    },
    "TransformResources": {
      "InstanceCount": 1,
      "InstanceType": "ml.m4.xlarge"
    },
    "TransformJobName": "sfn-binary-classification-prediction"
  },
  "Next": "ValidateOutput"
},
```

SageMaker 培訓 Job 示例

以下內容包括建Task立 Amazon SageMaker 訓練任務的狀態。

```
{
   "SageMaker CreateTrainingJob":{
      "Type":"Task",
      "Resource":"arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters":{
         "TrainingJobName":"search-model",
         "ResourceConfig":{
            "InstanceCount":4,
            "InstanceType":"ml.c4.8xlarge",
            "VolumeSizeInGB":20
         },
         "HyperParameters":{
            "mode":"batch_skipgram",
            "epochs":"5",
            "min_count":"5",
            "sampling_threshold":"0.0001",
            "learning_rate":"0.025",
            "window_size":"5",
            "vector_dim":"300",
            "negative_samples":"5",
            "batch_size":"11"
         },
         "AlgorithmSpecification":{
            "TrainingImage":"...",
            "TrainingInputMode":"File"
         },
         "OutputDataConfig":{
            "S3OutputPath":"s3://bucket-name/doc-search/model"
         },
         "StoppingCondition":{
            "MaxRuntimeInSeconds":100000
         },
         "RoleArn":"arn:aws:iam::123456789012:role/docsearch-stepfunction-iam-role",
         "InputDataConfig":[
            {
               "ChannelName":"train",
               "DataSource":{
                  "S3DataSource":{
                     "S3DataType":"S3Prefix",
                     "S3Uri":"s3://bucket-name/doc-search/interim-data/training-data/",
```

```
                    "S3DataDistributionType":"FullyReplicated"
                 }
              }
           }
        ]
     },
     "Retry":[
        {
           "ErrorEquals":[
              "SageMaker.AmazonSageMakerException"
           ],
           "IntervalSeconds":1,
           "MaxAttempts":100,
           "BackoffRate":1.1
        },
        {
           "ErrorEquals":[
              "SageMaker.ResourceLimitExceededException"
           ],
           "IntervalSeconds":60,
           "MaxAttempts":5000,
           "BackoffRate":1
        },
        {
           "ErrorEquals":[
              "States.Timeout"
           ],
           "IntervalSeconds":1,
           "MaxAttempts":5,
           "BackoffRate":1
        }
     ],
     "Catch":[
        {
           "ErrorEquals":[
              "States.ALL"
           ],
           "ResultPath":"$.cause",
           "Next":"Sagemaker Training Job Error"
        }
     ],
     "Next":"Delete Interim Data Job"
  }
}
```

SageMaker 標籤 Job 範例

以下內容包括建Task立 Amazon SageMaker 標籤任務的狀態。

```
{
  "StartAt": "SageMaker CreateLabelingJob",
  "TimeoutSeconds": 3600,
  "States": {
    "SageMaker CreateLabelingJob": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createLabelingJob.sync",
      "Parameters": {
        "HumanTaskConfig": {
          "AnnotationConsolidationConfig": {
            "AnnotationConsolidationLambdaArn": "arn:aws:lambda:us-
west-2:123456789012:function:ACS-TextMultiClass"
          },
          "NumberOfHumanWorkersPerDataObject": 1,
          "PreHumanTaskLambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:PRE-
TextMultiClass",
          "TaskDescription": "Classify the following text",
          "TaskKeywords": [
            "tc",
            "Labeling"
          ],
          "TaskTimeLimitInSeconds": 300,
          "TaskTitle": "Classify short bits of text",
          "UiConfig": {
            "UiTemplateS3Uri": "s3://s3bucket-example/TextClassification.template"
          },
          "WorkteamArn": "arn:aws:sagemaker:us-west-2:123456789012:workteam/private-
crowd/ExampleTesting"
        },
        "InputConfig": {
          "DataAttributes": {
            "ContentClassifiers": [
              "FreeOfPersonallyIdentifiableInformation",
              "FreeOfAdultContent"
            ]
          },
```

```
          "DataSource": {
            "S3DataSource": {
              "ManifestS3Uri": "s3://s3bucket-example/manifest.json"
            }
          }
        },
        "LabelAttributeName": "Categories",
        "LabelCategoryConfigS3Uri": "s3://s3bucket-example/labelcategories.json",
        "LabelingJobName": "example-job-name",
        "OutputConfig": {
          "S3OutputPath": "s3://s3bucket-example/output"
        },
        "RoleArn": "arn:aws:iam::123456789012:role/service-role/AmazonSageMaker-
ExecutionRole",
        "StoppingConditions": {
          "MaxHumanLabeledObjectCount": 10000,
          "MaxPercentageOfInputDatasetLabeled": 100
        }
      },
      "Next": "ValidateOutput"
    },
    "ValidateOutput": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.LabelingJobArn",
            "StringEquals": ""
          },
          "Next": "Succeed"
        }
      ],
      "Default": "Fail"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail",
      "Error": "InvalidOutput",
      "Cause": "Output is not what was expected. This could be due to a service outage
 or a misconfigured service integration."
    }
  }
```

```
}
```

SageMaker 處理 Job 範例

以下內容包括建Task立 Amazon SageMaker 處理任務的狀態。

```
{
  "StartAt": "SageMaker CreateProcessingJob Sync",
  "TimeoutSeconds": 3600,
  "States": {
    "SageMaker CreateProcessingJob Sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createProcessingJob.sync",
      "Parameters": {
        "AppSpecification": {
          "ImageUri": "737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-
learn:0.20.0-cpu-py3"
        },
        "ProcessingResources": {
          "ClusterConfig": {
            "InstanceCount": 1,
            "InstanceType": "ml.t3.medium",
            "VolumeSizeInGB": 10
          }
        },
        "RoleArn": "arn:aws:iam::123456789012:role/SM-003-
CreateProcessingJobAPIExecutionRole",
        "ProcessingJobName.$": "$.id"
      },
      "Next": "ValidateOutput"
    },
    "ValidateOutput": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.ProcessingJobArn",
            "StringEquals": ""
          },
          "Next": "Succeed"
        }
      ],
      "Default": "Fail"
```

```
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail",
      "Error": "InvalidConnectorOutput",
      "Cause": "Connector output is not what was expected. This could be due to a
 service outage or a misconfigured connector."
    }
  }
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 呼叫 Amazon SNS

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> 最佳化的 Amazon SNS 整合與 Amazon SNS AWS 開發套件整合有何不同
>
> 請求回應或等候傳回任務字符的回呼整合模式沒有最佳化。

支援的 Amazon SNS API：

> **Note**
>
> 「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

- `Publish`
  - 請求語法
  - 支援的參數
    - `Message`

- MessageAttributes
- MessageStructure
- PhoneNumber
- Subject
- TargetArn
- TopicArn

- 回應語法

> **Note**
>
> 中的參數Step Functions會在中表示 PascalCase，即使原生服務 API 位於駝峰中也是如此。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

以下內容包括發佈到 Amazon Simple Notification Service (Amazon SNS) 主題的`Task`狀態。

```
{
 "StartAt": "Publish to SNS",
 "States": {
   "Publish to SNS": {
     "Type": "Task",
     "Resource": "arn:aws:states:::sns:publish",
     "Parameters": {
       "TopicArn": "arn:aws:sns:us-east-1:123456789012:myTopic",
       "Message.$": "$.input.message",
       "MessageAttributes": {
         "my_attribute_no_1": {
           "DataType": "String",
           "StringValue": "value of my_attribute_no_1"
         },
         "my_attribute_no_2": {
           "DataType": "String",
           "StringValue": "value of my_attribute_no_2"
         }
       }
     },
     "End": true
    }
```

```
  }
}
```

傳遞動態值。您可以修改上面的示例以動態方式從此 JSON 有效負載傳遞屬性：

```
{
"input": {
  "message": "Hello world"
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

追加.$到該StringValue字段：

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

以下內容包括發佈到 Amazon SNS 主題，然後等待任務權杖傳回的Task狀態。請參閱[等候傳回任務字符的回呼](#)。

```
{
   "StartAt":"Send message to SNS",
   "States":{
      "Send message to SNS":{
         "Type":"Task",
         "Resource":"arn:aws:states:::sns:publish.waitForTaskToken",
         "Parameters":{
            "TopicArn":"arn:aws:sns:us-east-1:123456789012:myTopic",
            "Message":{
               "Input.$":"$",
               "TaskToken.$":"$$.Task.Token"
            }
```

```
        },
        "End":true
      }
   }
}
```

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

使用 Step Functions 呼叫 Amazon SQS

Step Functions 可以直接從Amazon States Language（ASL）控制某些 AWS 服務。如需了解詳細資訊，請參閱 使用其他 服務 和 將參數傳遞至服務 API。

> 最佳化的 Amazon SQS 整合與 Amazon SQS AWS 開發套件整合有何不同
>
> 請求回應或等候傳回任務字符的回呼整合模式沒有最佳化。

支援的 Amazon SQS API：

> Note
>
> 「Step Functions 數」中的工作有最大輸入或結果資料大小的配額。當您傳送至其他服務或從其他服務接收資料時，這會將您限制為 256 KB 的資料，做為 UTF-8 編碼字串。請參閱與狀態機器執行相關的配額。

- SendMessage

  支援的參數：

  - DelaySeconds
  - MessageAttribute
  - MessageBody
  - MessageDeduplicationId
  - MessageGroupId
  - QueueUrl

- Response syntax

> **Note**
>
> 中的參數Step Functions會以表示 PascalCase，即使原生服務 API 位於 camelCase 中。例如，您可以使用 Step Functions API 動作`startSyncExecution`並將其參數指定為`StateMachineArn`。

以下內容包括傳送 Amazon Simple Queue Service (Amazon SQS) 訊息的`Task`狀態。

```
{
 "StartAt": "Send to SQS",
 "States": {
   "Send to SQS": {
     "Type": "Task",
     "Resource": "arn:aws:states:::sqs:sendMessage",
     "Parameters": {
       "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
       "MessageBody.$": "$.input.message",
       "MessageAttributes": {
         "my_attribute_no_1": {
           "DataType": "String",
           "StringValue": "attribute1"
         },
         "my_attribute_no_2": {
           "DataType": "String",
           "StringValue": "attribute2"
         }
       }
     },
     "End": true
    }
  }
}
```

以下內容包括發佈到 Amazon SQS 佇列，然後等待任務權杖傳回的`Task`狀態。請參閱等候傳回任務字符的回呼。

```
{
   "StartAt":"Send message to SQS",
```

```
   "States":{
      "Send message to SQS":{
         "Type":"Task",
         "Resource":"arn:aws:states:::sqs:sendMessage.waitForTaskToken",
         "Parameters":{
            "QueueUrl":"https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
            "MessageBody":{
               "Input.$":"$",
               "TaskToken.$":"$$.Task.Token"
            }
         },
         "End":true
      }
   }
}
```

若要進一步了解如何在 Amazon SQS 中接收訊息，請參閱 Amazon 簡單佇列服務開發人員指南中的接收和刪除訊息。

如需Step Functions與其他 AWS 服務搭配使用時如何設定IAM權限的相關資訊，請參閱整合式服務的 IAM 政策。

以整合式服務 AWS Step Functions 的形式管理執行

Step Functions 與自己的 API 作為服務集成集成。這允許 Step Functions 直接從正在運行的執行的任務狀態啟動狀態機的新執行。在建置新的工作流程時，使用巢狀工作流程執行來降低主要工作流程的複雜性，並重複使用常見的程序。

> 優化 Step Functions 集成與步 Step Functions AWS SDK 集成有何不同
>
> - 整執行任務 (.sync)合模式可用。
>
> 請注意，請求回應或等候傳回任務字符的回呼整合模式沒有最佳化。

如需詳細資訊，請參閱下列內容：

- 從任務開始執行
- 使用其他 服務

- 將參數傳遞至服務 API

支援的 Step Functions 式 API 和語法：

- StartExecution
  - 請求語法
  - 支援的參數
    - Input
    - Name
    - StateMachineArn
  - 回應語法

以下包含 Task 狀態，此狀態會啟動另一個狀態機器的執行並等待其完成。

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution.sync:2",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

以下包含 Task 狀態，此狀態會啟動另一個狀態機器的執行。

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
```

```
      "Name":"ExecutionName"
   },
   "End":true
}
```

以下包含 Task 狀態，此狀態會實作回呼服務整合模式。

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution.waitForTaskToken",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!",
        "token.$": "$$.Task.Token"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

若要將巢狀工作流程執行與啟動它的父執行建立關聯，請傳遞特別命名的參數，其中包含從內容物件提取的執行 ID。啟動巢狀執行時，請使用名為 AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID 的參數。透過將 .$ 附加至參數名稱，並使用 $$.Execution.Id 參考內容物件中的 ID 來傳遞執行 ID。如需詳細資訊，請參閱 存取內容物件。

```
{
   "Type":"Task",
   "Resource":"arn:aws:states:::states:startExecution.sync",
   "Parameters":{
      "Input":{
        "Comment": "Hello world!",
        "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
       },
      "StateMachineArn":"arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld",
      "Name":"ExecutionName"
   },
   "End":true
}
```

巢狀狀態機器會傳回下列內容：

資源	輸出
startExecution.sync	字串
startExecution.sync:2	JSON

兩者都將等待巢狀狀態機器完成，但它們會傳回不同的 `Output` 格式。例如，如果您建立傳回物件的 Lambda 函數`{ "MyKey": "MyValue" }`，您會得到下列回應：

對於 startExecution.sync：

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

對於 startExecution.sync:2：

```
{
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

為巢狀狀態機器設定 IAM 許可

父狀態機器判斷子狀態機器是否已使用輪詢和事件完成執行。輪詢需要權限，`states:DescribeExecution`而傳送 EventBridge 至 Step Functions 的事件需要`events:PutTargets`、`events:PutRule`和的權限`events:DescribeRule`。如果您的 IAM 角色缺少這些許可，則父狀態機器可能會有一段延遲，才能知道子狀態機器的執行完成。

對於要求單一巢狀工作流程執行的狀態機器，請使用將權限限制`StartExecution`為該狀態機器的 IAM 政策。

如需詳細資訊，請參閱 Step Functions 的 IAM 許可。

呼叫第三方 API

HTTP 工作是一種任務狀態，可讓您呼叫工作流程中的任何公用協力廠商 API，例如 Salesforce 和 Stripe。若要呼叫第三方 API，請將 [工作] 狀態與`arn:aws:states:::http:invoke`資源搭配使用。然後，提供 API 端點配置詳細信息，例如 API URL，要使用的方法以及身份驗證詳細信息。

如果您使用工作流程 Studio 來建置包含 HTTP 工作的狀態機器，工作流程 Studio 會自動產生具有 HTTP 工作IAM原則的執行角色。如需詳細資訊，請參閱 在工作流程工作室中測試 HTTP 任務的角色。

主題

HTTP 任務定義

ASL 定義表示具有`http:invoke`資源的 HTTP 任務。以下 HTTP 任務定義調用條紋 API，該 API 返回所有客戶的列表。

```
"Call third-party API": {
  "Type": "Task",
  "Resource": "arn:aws:states:::http:invoke",
  "Parameters": {
    "ApiEndpoint": "https://api.stripe.com/v1/customers",
    "Authentication": {
      "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
    },
    "Method": "GET"
  },
```

```
  "End": true
}
```

「HTTP 任務」字段

HTTP 任務在其定義中包含以下字段。

`Resource` (必要)

若要指定工作類型，請在`Resource`欄位中提供其 ARN。對於 HTTP 工作，您可以依照下列方式指定`Resource`欄位。

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (必要)

包含`ApiEndpoint`、和`ConnectionArn`欄位`Method`，這些欄位提供您要呼叫的第三方 API 相關資訊。 `Parameters`也包含選擇性欄位，例如`Headers`和`QueryParameters`。

您可以`Parameters`在`Parameters`字段中指定靜態 JSON 和JsonPath語法的組合。如需詳細資訊，請參閱 將參數傳遞至服務 API。

`ApiEndpoint`(必填)

指定您要呼叫的第三方 API 的 URL。若要將查詢參數附加至 URL，請使用欄位`QueryParameters`。下面的例子顯示了如何調用 Stripe API 來獲取所有客戶的列表。

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

您也可以使用JsonPath語法指定參考路徑，以選取包含第三方 API URL 的 JSON 節點。例如，假設您想要使用特定的客戶 ID 呼叫 Stripe 的其中一個 API。想像一下，您已經提供了以下狀態輸入。

```
{
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

若要使用 Stripe API 擷取此客戶 ID 的詳細資訊，請指定`ApiEndpoint`如下列範例所示。此範例使用內建函數和參考路徑。

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

在運行時，Step Functions 解析的值ApiEndpoint如下。

```
https://api.stripe.com/v1/customers/1234567890
```

Method(必填)

指定您要用來呼叫第三方 API 的 HTTP 方法。您可以在 HTTP 任務中指定以下方法之一：獲取，POST，放置，刪除，補丁，選項或頭。

例如，若要使用 GET 方法，請依下列方式指定Method欄位。

```
"Method": "GET"
```

您也可以使用參考路徑在執行階段指定方法。例如 **"Method.$": "$.myHTTPMethod"**。

Authentication(必填)

包含指ConnectionArn定如何驗證第三方 API 呼叫的欄位。 Step Functions支援ApiEndpoint使用的連線資源指定的驗證Amazon EventBridge。

ConnectionArn(必填)

指定EventBridge連接 ARN。

HTTP 任務需要EventBridge 連接，該連接可以安全地管理 API 提供者的身份驗證憑據。連線會指定用於授權第三方 API 的授權類型和認證。使用連線可協助您避免在狀態機器定義中進行硬式編碼 (例如 API 金鑰) 的密碼。在連接中，您也可以指定HeadersQueryParameters、和RequestBody參數。

當您建立 EventBridge 連線時，您需要提供驗證詳細資料。如需 HTTP 工作驗證如何運作的詳細資訊，請參閱HTTP 工作的驗證。

下列範例顯示如何在 HTTP 工作定義中指定Authentication欄位。

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
```

```
}
```

`Headers` (選用)

為 API 端點提供其他內容和中繼資料。您可以將標頭指定為字串或 JSON 陣列。

您可以在EventBridge連線中指定標頭，以及 HTTP 工作中的`Headers`欄位。我們建議您不要在`Headers`欄位中包含驗證詳細資料給您的 API 提供者。我們建議您將這些詳細資料納入您的EventBridge連線中。

Step Functions會將您在EventBridge連線中指定的標頭新增至您在 HTTP 工作定義中指定的標頭。如果定義和連接中存在相同的標頭鍵，則Step Functions使用在連接中為這些標頭指定的對應值。EventBridge如需如何Step Functions執行資料合併的詳細資訊，請參閱〈〉合併EventBridge連接和 HTTP 任務定義數據。

下列範例會指定將包含在第三方 API 呼叫中的標頭：`content-type`。

```
"Headers": {
  "content-type": "application/json"
}
```

您也可以使用參考路徑在執行階段指定標頭。例如 **`"Headers.$": "$.myHTTPHeaders"`**。

Step Functions設定`User-Agent`、`Range`和`Host`標頭。 Step Functions根據您正在調用的 API 設置`Host`標題的值。以下是這些標頭的範例。

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

您不能在 HTTP 任務定義中使用以下標頭。如果您使用這些標頭，HTTP 工作會失敗並顯示`States.Runtime`錯誤。

- A-IM
- Accept-Charset
- Accept-Datetime
- 接受編碼
- 快取控制

- 連線
- Content-Encoding
- 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
- Referer
- Server
- TE
- 預告片
- Transfer-Encoding
- Upgrade
- Via
- 警告
- X-向前-*
- x-amz-*

- x-amzn-*

`QueryParameters` (選用)

在 API URL 的末尾插入索引鍵值配對。您可以將查詢參數指定為字串、JSON 陣列或 JSON 物件。 Step Functions當它調用第三方 API 時，自動對查詢參數進行 URL 編碼。

例如，假設您想要呼叫 Stripe API 來搜尋以美元 (USD) 進行交易的客戶。想像一下，您已經提供了以下內容`QueryParameters`作為狀態輸入。

```
"QueryParameters": {
  "currency": "usd"
}
```

在運行時，Step Functions 附加`QueryParameters`到 API URL，如下所示。

```
https://api.stripe.com/v1/customers/search?currency=usd
```

您也可以使用參考路徑在執行階段指定查詢參數。例如 **`"QueryParameters.$": "$.myQueryParameters"`**。

如果您已在EventBridge連線中指定查詢參數，請Step Functions將這些查詢參數新增至您在 HTTP Task 定義中指定的查詢參數。如果定義和連接中存在相同的查詢參數鍵，則會針對這些標頭Step Functions使用在EventBridge連接中指定的對應值。如需如何Step Functions執行資料合併的詳細資訊，請參閱〈〉合併EventBridge連接和 HTTP 任務定義數據。

`Transform` (選用)

包含`RequestBodyEncoding`和`RequestEncodingOptions`欄位。默認情況下，Step Functions將請求主體作為 JSON 數據發送到 API 端點。

如果您的 API 提供者接受`form-urlencoded`要求內文，請使用`Transform`欄位來指定要求主體的 URL 編碼。您還必須將標`content-type`題指定為`application/x-www-form-urlencoded`。 Step Functions然後自動對您的請求主體進行 URL 編碼。

`RequestBodyEncoding`

指定請求主體的 URL 編碼。您可以指定下列其中一個值：`NONE`或`URL_ENCODED`。

- `NONE`— HTTP 要求主體將會是`RequestBody`欄位的序列化 JSON。這是預設值。
- `URL_ENCODED`— HTTP 要求主體將是欄位的 URL 編碼表單資料。`RequestBody`

RequestEncodingOptions

如果設RequestBodyEncoding定為，則決定要用於要求主體中陣列的編碼選項URL_ENCODED。

Step Functions支援下列陣列編碼選項。如需這些選項及其範例的詳細資訊，請參閱在要求主體上套用 URL 編碼。

- INDICES— 使用陣列元素的索引值對陣列進行編碼。依預設，Step Functions會使用此編碼選項。
- REPEAT— 為陣列中的每個項目重複索引鍵。
- COMMAS— 將索引鍵中的所有值編碼為逗號分隔的值清單。
- BRACKETS— 重複陣列中每個項目的索引鍵，並將括號 [] 附加至索引鍵，以指出它是陣列。

下列範例會設定要求主體資料的 URL 編碼。它還指定在請求主體中使用數組的COMMAS編碼選項。

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
  }
}
```

RequestBody (選用)

接受您在狀態輸入中提供的 JSON 資料。在中RequestBody，您可以指定靜態 JSON 和JsonPath語法的組合。例如，假設您提供下列 state 輸入：

```
{
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

若要在執行階段ExpiryDate在要求主體中使用CardNumber和的這些值，您可以在要求主體中指定下列 JSON 資料。

```
"RequestBody": {
```

```
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

如果您要呼叫的第三方 API 需要要`form-urlencoded`求主體，您必須為要求主體資料指定 URL 編碼。如需詳細資訊，請參閱 在要求主體上套用 URL 編碼。

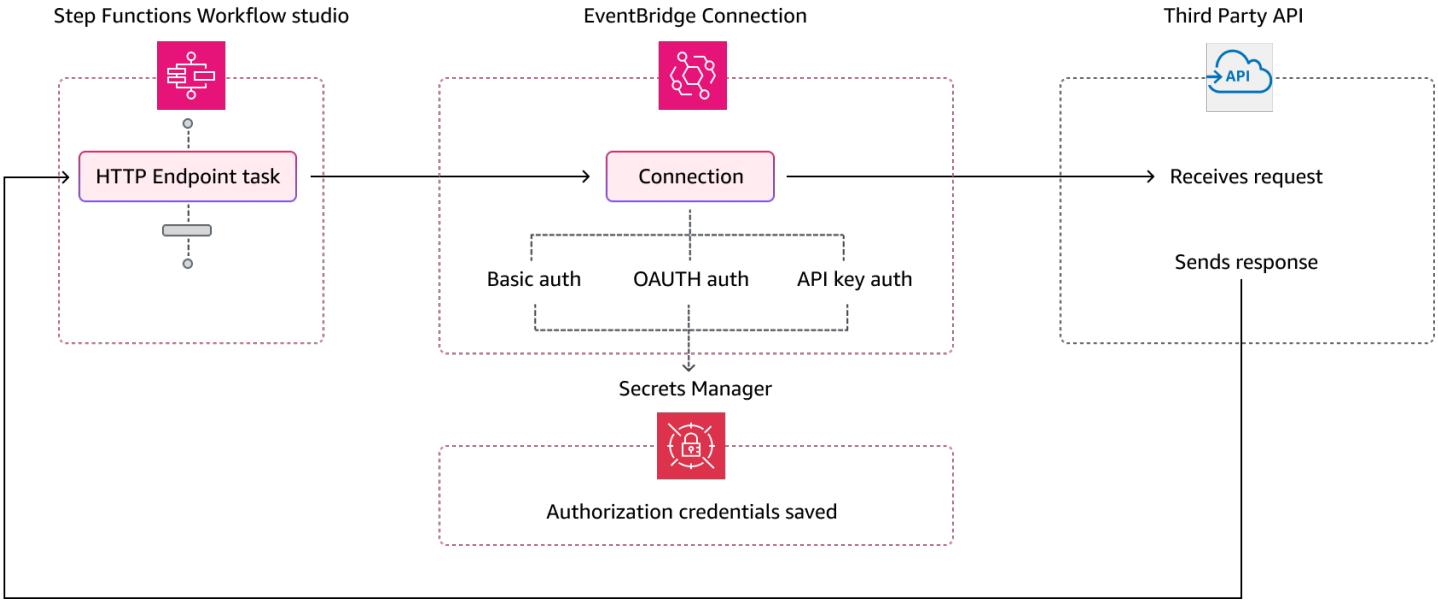
HTTP 工作的驗證

HTTP 任務需要EventBridge 連接，該連接可以安全地管理 API 提供者的身份驗證憑據。連線會指定用於授權第三方 API 的授權類型和認證。使用連線可協助您避免在狀態機器定義中進行硬式編碼 (例如 API 金鑰) 的密碼。 EventBridge 連線支援基本、OAuth 和 API 金鑰授權配置。

當您建立 EventBridge 連線時，您需要提供驗證詳細資料。您也可以包含 API 授權所需的標頭、內文和查詢參數。您必須在任何呼叫協力廠商 API 的 HTTP 工作中包含連線 ARN。

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。在這個秘密中，以加密的形式 EventBridge 存儲連接和授權參數。若要成功建立或更新連線，您必須使用具有 AWS 帳戶 使用 Secrets Manager 權限的連線。如需狀態機存取 EventBridge 連線所需IAM權限的詳細資訊，請參閱用於執行 HTTP 任務的 IAM 許可。

下圖顯示如何使用EventBridge連線Step Functions處理第三方 API 呼叫的驗證。



合併EventBridge連接和 HTTP 任務定義數據

當您呼叫 HTTP 工作時，您可以在EventBridge連線和 HTTP 工作定義中指定資料。此資料包括HeadersQueryParameters、和RequestBody參數。在呼叫協力廠商 API 之前，Step Functions會在所有情況下合併要求主體與連線主體參數，除非您的要求主體是字串且連線主體參數非空白。在此情況下，HTTP 工作會失敗並顯示States.Runtime錯誤。

如果 HTTP 工作定義和連線中指定了任何重複的索引鍵，會以EventBridge連線中的值Step Functions覆寫 HTTP 工作中的值。

下列清單說明如何在呼叫第三方 API 之前Step Functions合併資料：

- 標頭 — Step Functions 將您在連線中指定的任何標頭新增至 HTTP 工作Headers欄位中的標頭。如果標頭鍵之間存在衝突，則Step Functions使用連接中為這些標頭指定的值。例如，如果您在 HTTP 任務定content-type義和EventBridge連接中指定了標頭，則Step Functions會使用連接中指定的content-type標頭值。
- 查詢參數 — Step Functions 將您在連線中指定的任何查詢參數新增至 HTTP 作業QueryParameters欄位中的查詢參數。如果查詢參數鍵之間存在衝突，則會針對這些查詢參數Step Functions使用連線中指定的值。例如，如果您在 HTTP 任務定義和EventBridge連接中指定了maxItems查詢參數，則Step Functions會使用連接中指定的maxItems查詢參數值。
- 主體參數
  - Step Functions將連線中指定的任何要求主體值新增至 HTTP 工作RequestBody欄位中的要求主體值。如果要求主體索引鍵之間發生衝突，請Step Functions使用連線中指定的要求主體值。例如，假設您在 HTTP 任務Mode定義和EventBridge連接RequestBody的中指定了一個字段。 Step Functions會使用您在連線中指定的Mode欄位值。
  - 如果您將請求主體指定為字符串而不是 JSON 對象，並且EventBridge連接也包含請求主體，則Step Functions無法合併在這兩個位置指定的請求主體。它使 HTTP 任務失敗並顯示States.Runtime錯誤。

  Step Functions應用所有轉換，並在完成請求主體的合併後序列化請求主體。

下列範例會設定 Headers HTTP 工作和EventBridge連線中的QueryParameters、和RequestBody欄位。

HTTP 任務定義

```
{
  "Comment": "Data merging example for HTTP Task and EventBridge connection",
```

```
  "StartAt": "ListCustomers",
  "States": {
    "ListCustomers": {
      "Type": "Task",
      "Resource": "arn:aws:states:::http:invoke",
      "Parameters": {
        "Authentication": {
          "ConnectionArn": "arn:aws:events:us-
east-1:123456789012:connection/Example/81210c42-8af1-456b-9c4a-6ff02fc664ac"
        },
        "ApiEndpoint": "https:/example.com/path",
        "Method": "GET",
        "Headers": {
          "Request-Id": "my_request_id",
          "Header-Param": "state_machine_header_param"
        },
        "RequestBody": {
          "Job": "Software Engineer",
          "Company": "AnyCompany",
          "BodyParam": "state_machine_body_param"
        },
        "QueryParameters": {
          "QueryParam": "state_machine_query_param"
        }
      }
    }
  }
}
```

EventBridge 連線

```
{
  "AuthorizationType": "API_KEY",
  "AuthParameters": {
    "ApiKeyAuthParameters": {
      "ApiKeyName": "ApiKey",
      "ApiKeyValue": "key_value"
    },
    "InvocationHttpParameters": {
      "BodyParameters": [
        {
          "Key": "BodyParam",
          "Value": "connection_body_param"
```

```
        }
      ],
      "HeaderParameters": [
        {
          "Key": "Header-Param",
          "Value": "connection_header_param"
        }
      ],
      "QueryStringParameters": [
        {
          "Key": "QueryParam",
          "Value": "connection_query_param"
        }
      ]
    }
  }
}
```

在此範例中，在 HTTP 工作和EventBridge連線中指定重複的索引鍵。因此，會使用連線中的值Step Functions覆寫 HTTP 工作中的值。下列程式碼片段會顯示Step Functions傳送至協力廠商 API 的 HTTP 要求。

```
POST /path?QueryParam=connection_query_param HTTP/1.1
Apikey: key_value
Content-Length: 79
Content-Type: application/json; charset=UTF-8
Header-Param: connection_header_param
Host: example.com
Range: bytes=0-262144
Request-Id: my_request_id
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1

{"Job":"Software Engineer","Company":"AnyCompany","BodyParam":"connection_body_param"}
```

在要求主體上套用 URL 編碼

默認情況下，Step Functions將請求主體作為 JSON 數據發送到 API 端點。如果您的第三方 API 提供者需要要form-urlencoded求主體，您必須為要求主體指定 URL 編碼。 Step Functions然後根據您選取的 URL 編碼選項，自動對要求主體進行 URL 編碼。

您可以使用欄位Transform指定 URL 編碼。此欄位包含指定是否要為要求主體套用 URL 編碼的RequestBodyEncoding欄位。當您指定RequestBodyEncoding欄位時，請先Step Functions

將 JSON 要求內文轉換為要form-urlencoded求內文，再呼叫第三方 API。您也必須將content-type標頭指定為，application/x-www-form-urlencoded因為接受 URL 編碼資料的 API 需要標content-type頭。

若要在要求主體中編碼陣列，請Step Functions提供下列陣列編碼選項。

- INDICES— 重複陣列中每個項目的索引鍵，並將括號 [] 附加至索引鍵，以指出它是陣列。此括號包含陣列元素的索引。添加索引可幫助您指定數組元素的順序。依預設，Step Functions會使用此編碼選項。

  例如，如果您的請求主體包含以下數組。

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions將此數組編碼為以下字符串。

  ```
  array[0]=a&array[1]=b&array[2]=c&array[3]=d
  ```

- REPEAT— 為陣列中的每個項目重複索引鍵。

  例如，如果您的請求主體包含以下數組。

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions將此數組編碼為以下字符串。

  ```
  array=a&array=b&array=c&array=d
  ```

- COMMAS— 將索引鍵中的所有值編碼為逗號分隔的值清單。

  例如，如果您的請求主體包含以下數組。

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions將此數組編碼為以下字符串。

  ```
  array=a,b,c,d
  ```

- BRACKETS— 重複陣列中每個項目的索引鍵，並將括號 [] 附加至索引鍵，以指出它是陣列。

例如，如果您的請求主體包含以下數組。

```
{"array": ["a","b","c","d"]}
```

Step Functions將此數組編碼為以下字符串。

```
array[]=a&array[]=b&array[]=c&array[]=d
```

用於執行 HTTP 任務的 IAM 許可

您的狀態機器執行角色必須具有 HTTP 工作的`states:InvokeHTTPEndpoint`和`events:RetrieveConnectionCredentials`和`secretsmanager:GetSecretValue`和`secretsmanager:DescribeSecret`權限，才能呼叫第三方 API。以下 IAM 政策示例授予您的狀態機器角色調用 Stripe API 所需的最低權限。此 IAM 政策還授予狀態機器角色存取特定 EventBridge 連線的權限，包括存放在 Secrets Manager 中的此連線的密碼。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": "states:InvokeHTTPEndpoint",
            "Resource": "arn:aws:states:us-
east-2:123456789012:stateMachine:myStateMachine",
            "Condition": {
                "StringEquals": {
                    "states:HTTPMethod": "GET"
                },
                "StringLike": {
                    "states:HTTPEndpoint": "https://api.stripe.com/*"
                }
            }
        },
        {
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
                "events:RetrieveConnectionCredentials",
```

```
            ],
            "Resource": "arn:aws:events:us-
east-2:123456789012:connection/oauth_connection/aeabd89e-d39c-4181-9486-9fe03e6f286a"
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:events!connection/*"
        }
    ]
}
```

HTTP 工作範例

下列狀態機器定義顯示包含Headers、QueryParametersTransform、和RequestBody參數的 HTTP 工作。HTTP 任務調用條紋 API，https://api.stripe.com/v1/invoices，以生成一個發票。HTTP 工作也會使用編碼選項指定要求主體的 URL INDICES 編碼。

請確定您已建立EventBridge連線。下面的例子顯示了使用 BASIC 身份驗證類型創建的連接。

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

請記住將##文本替換為特定於資源的信息。

```
{
  "Comment": "A state machine that uses HTTP Task",
  "StartAt": "CreateInvoiceAPI",
  "States": {
    "CreateInvoiceAPI": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:states:::http:invoke",
      "Parameters": {
        "ApiEndpoint": "https://api.stripe.com/v1/invoices",
        "Method": "POST",
        "Authentication": {
          "ConnectionArn": ""arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
        },
        "Headers": {
          "Content-Type": "application/x-www-form-urlencoded"
        },
        "RequestBody": {
          "customer.$": "$.customer_id",
          "description": "Monthly subscription",
          "metadata": {
            "order_details": "monthly report data"
          }
        },
        "Transform": {
          "RequestBodyEncoding": "URL_ENCODED",
          "RequestEncodingOptions": {
            "ArrayFormat": "INDICES"
          }
        }
      },
      "Retry": [
        {
          "ErrorEquals": [
            "States.Http.StatusCode.429",
            "States.Http.StatusCode.503",
            "States.Http.StatusCode.504",
            "States.Http.StatusCode.502"
          ],
          "BackoffRate": 2,
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "JitterStrategy": "FULL"
        }
      ],
      "Catch": [
        {
          "ErrorEquals": [
            "States.Http.StatusCode.404",
            "States.Http.StatusCode.400",
```

```
            "States.Http.StatusCode.401",
            "States.Http.StatusCode.409",
            "States.Http.StatusCode.500"
          ],
          "Comment": "Handle all non 200 ",
          "Next": "HandleInvoiceFailure"
        }
      ],
      "End": true
    }
  }
}
```

若要執行此狀態機器，請提供客戶 ID 作為輸入，如下列範例所示：

```
{
    "customer_id": "1234567890"
}
```

下面的例子顯示了Step Functions發送到條紋 API 的 HTTP 請求。

```
POST /v1/invoices HTTP/1.1
Authorization: Basic <base64 of username and password>
Content-Type: application/x-www-form-urlencoded
Host: api.stripe.com
Range: bytes=0-262144
Transfer-Encoding: chunked
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1

description=Monthly%20subscription&metadata%5Border_details%5D=monthly%20report
%20data&customer=1234567890
```

測試一個 HTTP 任務

您可以透過主控台、SDK 或使用 TestStateAPI AWS CLI 來測試 HTTP 工作。下列程序說明如何在Step Functions主控台中使用 TestState API。您可以反覆測試 API 要求、回應和驗證詳細資料，直到 HTTP 工作如預期般運作為止。

在主控台中測試 HTTP 工作Step Functions狀態

1. 開啟「Step Functions」主控台。

2. 選擇建立狀態機以開始建立狀態機，或選擇包含 HTTP 工作的現有狀態機器。

   如果要在現有狀態機器中測試任務，請參閱步驟 4。

3. 在工作流程設計模式工作室中，以視覺化方式設定 HTTP 工作。或者選擇「程式碼」模式，從本機開發環境中複製貼上狀態機器定義。

4. 在 [設計] 模式中，選擇 [工作流程 Studio] 面Inspector板中的 [測試狀態]。

5. 在「測試狀態」對話方塊中，執行下列操作：

   a. 對於執行角色，請選擇要測試狀態的執行角色。如果您沒有具有 HTTP 任務足夠權限的角色，請參在工作流程工作室中測試 HTTP 任務的角色閱建立角色。

   b. (選擇性) 提供測試所選狀態所需的任何 JSON 輸入。

   c. 對於「檢驗層級」，請保留 INFO 的預設選項。此級別顯示 API 調用的狀態和狀態輸出。這對於快速檢查 API 響應非常有用。

   d. 選擇 [開始測試]。

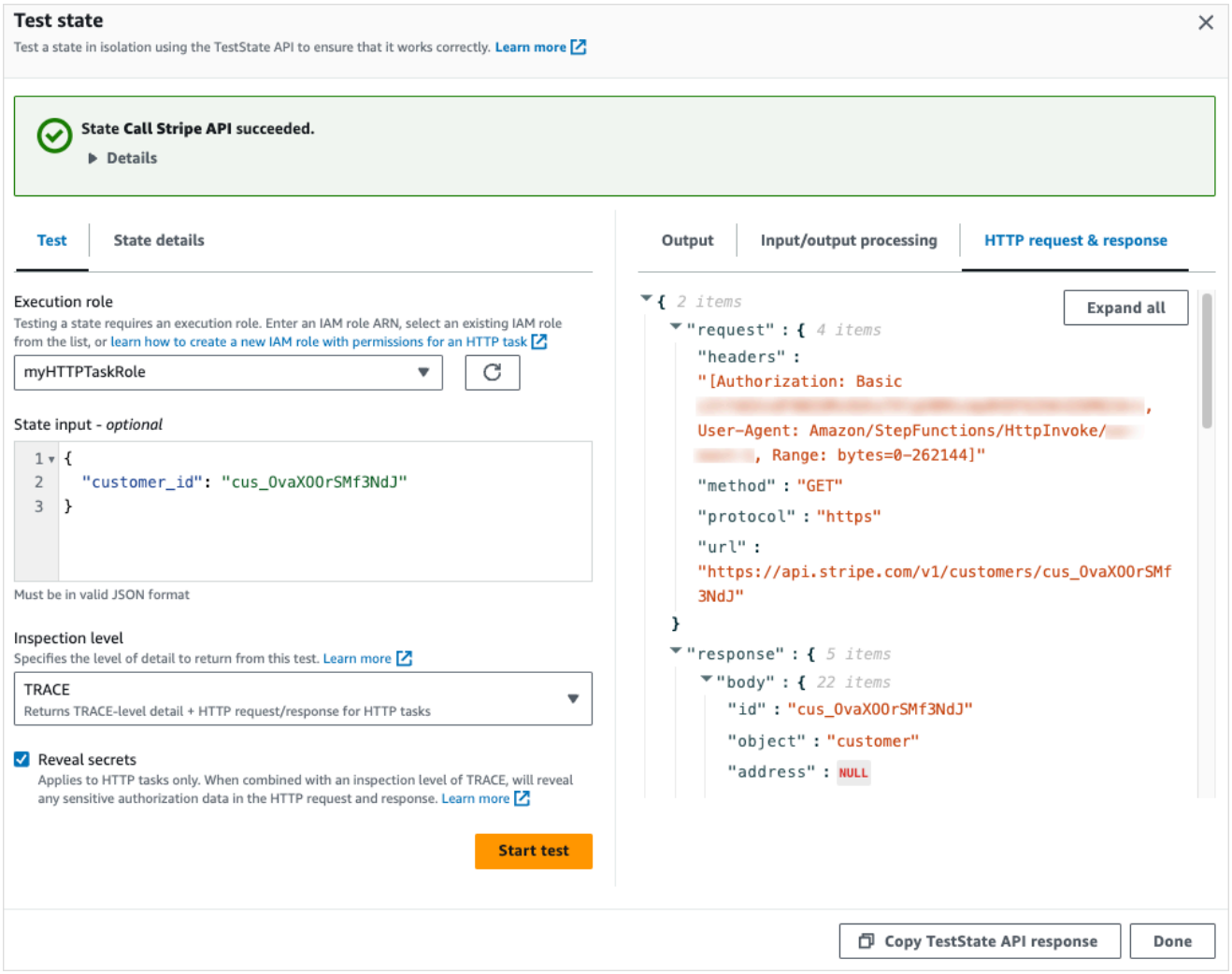
   e. 如果測試成功，狀態輸出會顯示在「測試狀態」對話方塊的右側。如果測試失敗，就會出現錯誤。

      在對話方塊的 [狀態詳細資料] 索引標籤中，您可以看到狀態定義和EventBridge連線的連結。

   f. 將「檢驗層級」變更為「追蹤」。此級別顯示原始 HTTP 請求和響應，並且對於驗證標頭，查詢參數和其他 API 特定的詳細信息非常有用。

   g. 選擇「揭露秘密」核取方塊。與 TRACE 結合使用時，此設定可讓您查看EventBridge連線所插入的敏感資料，例如 API 金鑰。您用來存取主控台的使用IAM者身分必須具有執行`states:RevealSecrets`動作的權限。如果沒有此權限，則會在您啟動測試時Step Functions擲回拒絕存取錯誤。如需設定`states:RevealSecrets`權限的IAM原則範例，請參閱IAM使用 TestState API 的權限。

      下圖顯示成功之 HTTP 工作的測試。此狀態的 [檢驗層級] 設定為 [TRACE]。下圖中的 HTTP 要求與回應索引標籤會顯示第三方 API 呼叫的結果。



h. 選擇 [開始測試]。

i. 如果測試成功，您可以在 HTTP 請求和響應選項卡下看到您的 HTTP 詳細信息。

不支援的 HTTP 工作回應

如果傳回的回應符合下列條件之一，HTTP 工作會失敗並顯示`States.Runtime`錯誤：

- 回應包含`application/octet-stream`、`image/*`、或的內容類型標頭。`video/*` `audio/*`
- 響應不能作為有效字符串讀取。例如，二進位或影像資料。

服務整合模式

AWS Step Functions 直接與 Amazon 州語言中的服務整合。您可以使用三種服務整合模式來控制這些 AWS 服務。

- 呼叫服務，讓 Step Functions 式在取得 HTTP 回應後立即進入下一個狀態。
- 呼叫服務並讓 Step Functions 等待工作完成。
- 使用任務令牌調用服務，並讓 Step Functions 等待，直到該令牌與有效負載一起返回。

這些服務整合模式中的每一種都是由您在任務定義"Resource"欄位中建立 URI 的方式所控制。

呼叫整合服務的方式

- 請求回應
- 執行任務 (.sync)
- 等候傳回任務字符的回呼

如需為整合式服務設定 AWS Identity and Access Management (IAM) 的相關資訊，請參閱整合式服務的 IAM 政策。

請求回應

當您在工作狀態的"Resource"字串中指定服務，而且只提供資源時，Step Functions 會等待 HTTP 回應，然後進入下一個狀態。Step Functions 不會等待工作完成。

以下範例顯示如何發佈 Amazon SNS 主題。

```
"Send message to SNS":{
   "Type":"Task",
   "Resource":"arn:aws:states:::sns:publish",
   "Parameters":{
      "TopicArn":"arn:aws:sns:us-east-1:123456789012:myTopic",
      "Message":"Hello from Step Functions!"
   },
   "Next":"NEXT_STATE"
}
```

此範例參考 Amazon SNS 的發佈 API。此工作流程會在呼叫 Publish API 之後繼續進行下一個狀態。

> **Tip**
>
> 若要將使用要求回應服務整合模式的範例工作流程部署至您的工作流程 AWS 帳戶，請參閱單元 2- AWS Step Functions 研討會的要求回應。

執行任務 (.sync)

對於整合式服務 (例如 AWS Batch 和 Amazon ECS)，Step Functions 可以等待請求完成，然後再進入下一個狀態。若要讓 Step Functions 式等候，請在任務狀態定義中指定"Resource"欄位，並在資源 URI 後面加上.sync尾碼。

例如，提交 AWS Batch 工作時，請使用狀態機定義中的"Resource"欄位，如此範例所示。

```
"Manage Batch task": {
  "Type": "Task",
  "Resource": "arn:aws:states:::batch:submitJob.sync",
  "Parameters": {
    "JobDefinition": "arn:aws:batch:us-east-2:123456789012:job-definition/
testJobDefinition",
    "JobName": "testJob",
    "JobQueue": "arn:aws:batch:us-east-2:123456789012:job-queue/testQueue"
  },
  "Next": "NEXT_STATE"
}
```

將.sync部分附加到資源 Amazon 資源名稱（ARN）表示 Step Functions 等待任務完成。呼叫 AWS Batch submitJob 之後，工作流程會暫停。當工作完成時，Step Functions 進展到下一個狀態。如需詳細資訊，請參閱 AWS Batch 範例專案：管理批次工作 (AWS Batch、Amazon SNS)。

如果使用 this (.sync) 服務整合模式的工作中止，而 Step Functions 無法取消工作，則整合式服務可能會產生額外費用。在下列情況下，工作可以中止：

- 狀態機執行已停止。
- 平行狀態的不同分支失敗，並顯示未捕獲的錯誤。
- Map 狀態的迭代失敗，並顯示未捕獲的錯誤。

Step Functions 將盡最大努力取消任務。例如，如果一個 Step Functions states:startExecution.sync 任務被中止，它將調用 Step Functions StopExecution API 動作。但是，Step Functions 可能無法取消工作。這種情況的原因包括但不限於：

- 您的 IAM 執行角色缺乏進行對應 API 呼叫的權限。
- 發生暫時性服務中斷。

當您使用.sync服務整合模式時，Step Functions 會使用耗用您指派的配額和事件的輪詢來監視工作的狀態。對於相同帳戶內的.sync呼叫，Step Functions 會使用 EventBridge 事件並輪詢您在狀態中指定的 API。Task對於跨帳戶.sync調用，Step Functions 僅使用輪詢。例如，「Step Functions」會對DescribeExecutionAPI 執行輪詢states:StartExecution.sync，並使用您指派的配額。

> **Tip**
>
> 若要將使用執行作業 (.sync) 服務整合模式的範例 Job 流程部署到您的 Job 流程 AWS 帳戶，請參閱研討會的單元 3-執行作業 (.sync)。 AWS Step Functions

若要查看支援等候工作完成的整合服務清單 (.sync) 的詳細資訊，請參閱Step Functions 的最佳化整合。

> **Note**
>
> 使用該.sync模式的服務整合需要額外的 IAM 許可。如需詳細資訊，請參閱 整合式服務的 IAM 政策。

在某些情況下，您可能希望 Step Functions 在工作完全完成之前繼續您的工作流程。您可以使用與使用等候傳回任務字符的回呼服務整合模式相同的方式來達成此目的。為此，請將任務令牌傳遞給您的工作，然後使用SendTaskSuccess或 SendTaskFailureAPI 調用將其返回。Step Functions 將使用您在該呼叫中提供的資料來完成工作、停止監視工作，並繼續工作流程。

等候傳回任務字符的回呼

回呼任務可讓工作流程暫停，直到任務字符傳回。任務可能需要等候人員核准、與第三方進行整合，或者呼叫舊版系統。對於這類工作，您可以暫停 Step Functions，直到工作流程執行達到一年的服務配額為止 (請參閱，與狀態節流有關的配額)，然後等待外部處理序或工作流程完成。在這些情況下，Step Functions 允許您將任務令牌傳遞給 AWS SDK 服務集成，以及一些優化的服務集成。這時任務會暫停等候，直到其收到 SendTaskSuccess 或 SendTaskFailure 呼叫傳回的任務字符。

如果使用回調任務令牌的Task狀態超時，則會生成新的隨機令牌。您可以從上下文對象訪問任務令牌。

> **Note**
>
> 工作權杖必須至少包含一個字元，且不得超過 1024 個字元。

若要.waitForTaskToken搭配 AWS SDK 整合使用，您使用的 API 必須具有可放置工作權杖的參數欄位。

> **Note**
>
> 您必須從相同 AWS 帳戶內的主體傳遞工作權杖。如果您從不同 AWS 帳戶中的主體發送令牌，則令牌將不起作用。

> **Tip**
>
> 要將使用回調任務令牌服務集成模式的示例工作流程部署到您的 AWS 帳戶，請參閱模塊 4-使用 AWS Step Functions 工作坊的任務令牌等待回調。

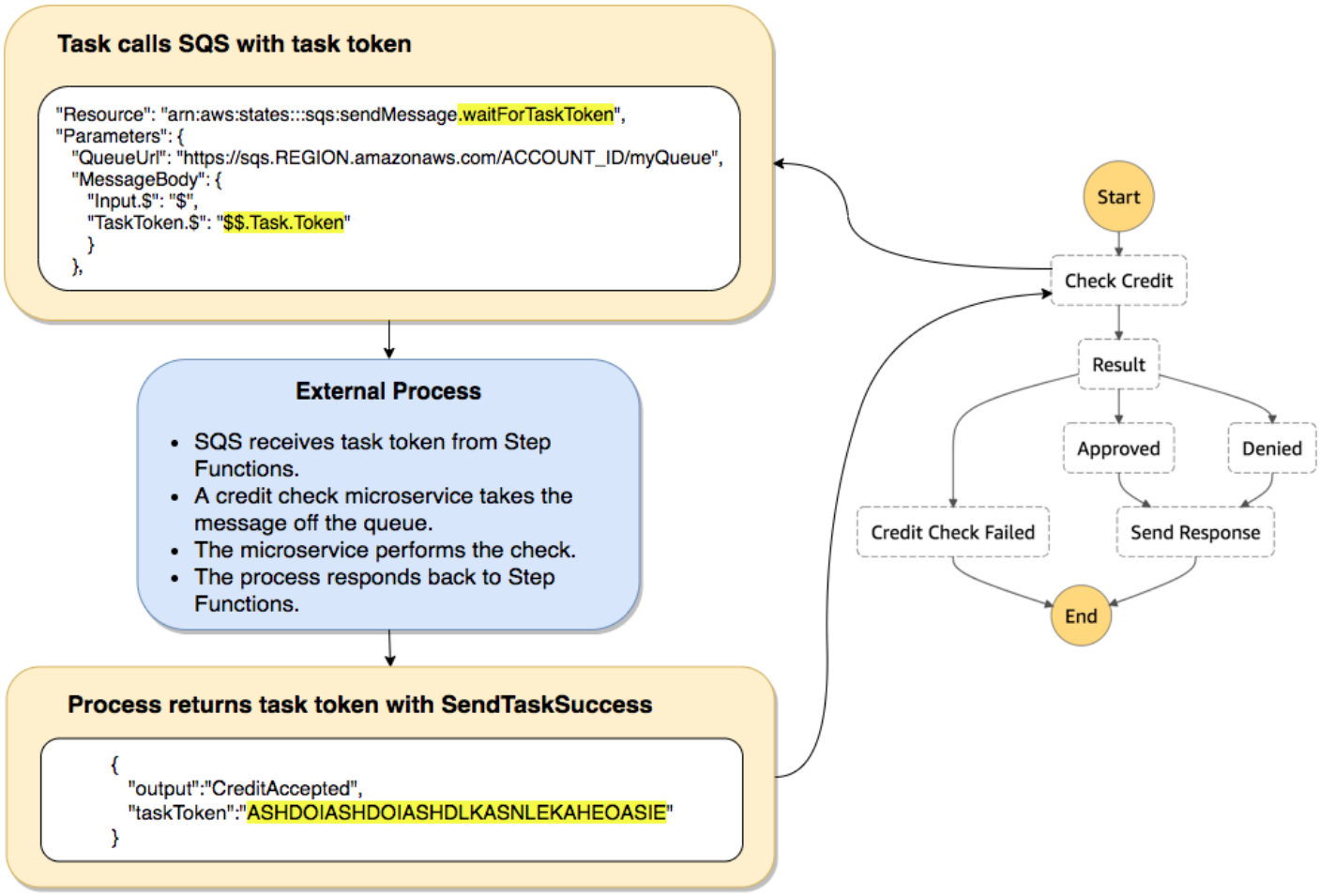
若要查看支援等候任務字符的整合服務清單 (.waitForTaskToken) 的詳細資訊，請參閱Step Functions 的最佳化整合。

主題

- 任務字符範例
- 取得來自內容物件的字符
- 設定活動訊號逾時為正在等候的任務

任務字符範例

在此範例中，「Step Functions」工作流程需要與外部微服務整合，才能在核准工作流程中執行信用檢查。Step Functions 會發佈 Amazon SQS 訊息，其中包含任務權杖做為訊息的一部分。外部系統與 Amazon SQS 整合，並將訊息從佇列中取出。完成後，它將返回結果和原始任務令牌。Step Functions 接著會繼續其工作流程。



參考 Amazon SQS 的任務定義"Resource"欄位.waitForTaskToken會附加在結尾。

```
"Send message to SQS": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
  "Parameters": {
    "QueueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/myQueue",
    "MessageBody": {
        "Message": "Hello from Step Functions!",
        "TaskToken.$": "$$.Task.Token"
     }
  },
  "Next": "NEXT_STATE"
}
```

這告訴 Step Functions 暫停並等待任務令牌。當您使用 .waitForTaskToken 指定資源時，使用包含特殊路徑目標 ($$.Task.Token) 之狀態定義的 "Parameters" 欄位，就能存取該任務字符。最初的 $$. 會指定該路徑要存取內容物件，並取得進行中執行之目前任務的任務字符。

完成之後，外部服務會呼叫 SendTaskSuccess 或 SendTaskFailure，其中包含 taskToken。工作流程只有在此時才會繼續進入下一個狀態。

> **Note**
>
> 若要避免當程序無法隨著 SendTaskSuccess 或 SendTaskFailure 傳送任務字符而造成無限期等候的情況，請參閱設定活動訊號逾時為正在等候的任務。

取得來自內容物件的字符

內容物件是一種 JSON 物件，其中包含關於執行的資訊。正如狀態輸入，它可以在執行期間搭配路徑，從 "Parameters" 欄位中存取而得。從任務定義存取時，它會包含有關特定執行的資訊，包括任務字符。

```
{
    "Execution": {
        "Id": "arn:aws:states:us-
east-1:123456789012:execution:stateMachineName:executionName",
        "Input": {
           "key": "value"
        },
        "Name": "executionName",
        "RoleArn": "arn:aws:iam::123456789012:role...",
        "StartTime": "2019-03-26T20:14:13.192Z"
    },
    "State": {
        "EnteredTime": "2019-03-26T20:14:13.192Z",
        "Name": "Test",
        "RetryCount": 3
    },
    "StateMachine": {
        "Id": "arn:aws:states:us-east-1:123456789012:stateMachine:stateMachineName",
        "Name": "name"
    },
    "Task": {
        "Token": "h7XRiCdLtd/83p1E0dMccoxlzFhglsdkzpK9mBVKZsp7d9yrT1W"
```

```
    }
}
```

您可以使用特殊路徑，從任務定義之 "Parameters" 欄位中存取任務字符。若要存取輸入或內容物件，您必須先指定該參數將作為路徑，方法是將 .$ 附加到參數名稱。以下命令會採用 "Parameters" 規格，指定來自輸入和內容物件的節點。

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

在這兩種情況下，附加.$到參數名稱告訴 Step Functions 數期望路徑。在第一個案例中，"$" 就是包括整個輸入的路徑。在第二個案例中，$$. 會指定路徑將存取內容物件，而 $$.Task.Token 會將參數設定成正在執行中內容物件的任務字符值。

在 Amazon SQS 範例.waitForTaskToken中，在"Resource"欄位中告知 Step Functions 等待任務權杖傳回。該"TaskToken.$": "**$$.Task.Token**"參數會將該權杖做為 Amazon SQS 訊息的一部分傳遞。

```
"Send message to SQS": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
  "Parameters": {
    "QueueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/myQueue",
    "MessageBody": {
        "Message": "Hello from Step Functions!",
        "TaskToken.$": "$$.Task.Token"
     }
  },
  "Next": "NEXT_STATE"
}
```

如需詳細資訊，請參閱本指南中輸入和輸出處理一節的內容物件。

設定活動訊號逾時為正在等候的任務

正在等候任務字符的任務將會等候，直到執行到達一年的服務配額 (請參閱與狀態節流有關的配額)。為了避免執行發生停滯，您可以設定在狀態機器定義中設定活動訊號逾時間隔。使用 HeartbeatSeconds 欄位，指定逾時的間隔。

```
{
  "StartAt": "Push to SQS",
  "States": {
    "Push to SQS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
      "HeartbeatSeconds": 600,
      "Parameters": {
        "MessageBody": { "myTaskToken.$": "$$.Task.Token" },
        "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/push-based-queue"
      },
      "ResultPath": "$.SQS",
      "End": true
    }
  }
}
```

在此狀態機器定義中，任務會將訊息推送至 Amazon SQS，並等待外部程序使用提供的任務權杖回呼。"HeartbeatSeconds": 600 欄位會將活動訊號逾時間隔設定為 10 分鐘。此任務將等候下列其中一個 API 動作傳回任務字符：

- SendTaskSuccess
- SendTaskFailure
- SendTaskHeartbeat

如果等候任務沒有在 10 分鐘期間內收到有效的任務字符，則此任務就會失敗，且收到名稱 States.Timeout 的錯誤。

如需詳細資訊，請參閱回呼任務範例專案：回呼模式範例 (Amazon SQS、Amazon SNS、Lambda)。

將參數傳遞至服務 API

在 Task 狀態中使用 Parameters 欄位，以控制將哪些參數傳遞到服務 API。

在Parameters欄位內，您必須在 API 動作中使用陣列參數的複數形式。例如，如果您使用 DescribeSnapshots API 動作的篩選器欄位與 Amazon EC2 整合，則必須將欄位定義為Filters。如果您不使用複數形式，步驟函數會傳回下列錯誤：

```
The field Filter is not supported by Step Functions.
```

將靜態 JSON 作為參數傳遞

您可以直接在狀態機器定義中包含 JSON 物件，以當作參數傳遞到資源。

例如，若要為 AWS Batch 的SubmitJob API 設定 RetryStrategy 參數，您可以將下列納入您的參數。

```
"RetryStrategy": {
  "attempts": 5
}
```

您也可以隨著靜態 JSON 傳遞多個參數。作為更完整的範例，以下是發佈到名為的 Amazon SNS 主題的任務規格的Resource和Parameters欄位*myTopic*。

```
"Resource": "arn:aws:states:::sns:publish",
  "Parameters": {
     "TopicArn": "arn:aws:sns:us-east-2:123456789012:myTopic",
     "Message": "test message",
     "MessageAttributes": {
       "my attribute no 1": {
         "DataType": "String",
         "StringValue": "value of my attribute no 1"
       },
       "my attribute no 2": {
         "DataType": "String",
         "StringValue": "value of my attribute no 2"
       }
     }
  },
```

使用路徑將狀態輸入作為參數傳遞

您可以使用路徑將狀態輸入的部分作為參數傳遞。路徑是一個字符串，以開頭$，用於標識 JSON 文本中的組件。步驟函數路徑使用JsonPath語法。

若要指定參數使用路徑，請以結束參數名稱.$。例如，如果您的狀態輸入包含名為的節點內的文字message，您可以使用路徑將該文字當做參數傳遞。

考慮下面的狀態輸入：

```
{
```

```
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

要傳遞命名message為參數的節點的值，請指定以下語法：

```
"Parameters": {"myMessage.$": "$.input.message"},
```

步驟函數然後將值foo作為參數傳遞。

如需有關在步驟函數中使用參數的詳細資訊，請參閱下列內容：

- 輸入和輸出處理
- InputPath、參數和 ResultSelector

依參數方式傳遞內容物件節點

除了靜態內容和來自狀態輸入的節點，您還可以依參數方式來傳遞內容物件的節點。內容物件是一種會在狀態機器執行期間存在的動態 JSON 資料。它包含有關您的狀態機器和目前執行的資訊。您可以使用狀態定義之 "Parameters" 欄位的路徑，存取內容物件。

如需關於內容物件及如何存取 "Parameters" 欄位資料的詳細資訊，請參閱下列：

- 內容物件
- 存取內容物件
- 取得來自內容物件的字符

變更支援 AWS SDK 整合的記錄

下表摘要說明服務最初與 Step Functions 整合的時間，以及其整合 API 最近更新的時間。如需使用整合的詳細資訊，請參閱AWS SDK 服務整合。

> **Important**
>
> API 操作支持按季度發布。已支援的動作 (例如新參數) 的更新可能無法立即使用。

服務	初步支援	Updated	
AWS AppFabric	2024年1月18日		
B2B Data Interchange	2024年1月18日		
AWS 資料匯出	2024年1月18日		
Amazon Bedrock	2024年1月18日		
Amazon Bedrock Agents	2024年1月18日		
Amazon Bedrock Runtime Agents	2024年1月18日		
Amazon Bedrock Runtime	2024年1月18日		
Amazon CloudFront KeyValueStore	2024年1月18日		
Amazon CodeGuru Security	2024年1月18日		
AWS 成本最佳化中心	2024年1月18日		
Amazon DataZone	2024年1月18日		
Amazon EKS Auth	2024年1月18日		
AWS Entity Resolution	2024年1月18日		
AWS 免費方案	2024年1月18日		

服務	初步支援	Updated	
Amazon Inspector Scan	2024年1月18日		
AWS Launch Wizard	2024年1月18日		
Amazon Managed Blockchain Query	2024年1月18日		
AWS Elemental MediaPackage V2	2024年1月18日		
AWS HealthImaging	2024年1月18日		
Network Manager	2024年1月18日		
AWS Payment Cryptography	2024年1月18日		
AWS Payment Cryptography Data	2024年1月18日		
AWS Private CA Connector for Active Directory	2024年1月18日		
Amazon Q Business	2024年1月18日		
Amazon Q Connect	2024年1月18日		
AWS re:Post	2024年1月18日		
Amazon Timestream Query	2024年1月18日		
Amazon Timestream Write	2024年1月18日		
Trusted Advisor	2024年1月18日		

服務	初步支援	Updated	
Verified Permissions	2024年1月18日		
Amazon WorkSpaces Thin Client	2024年1月18日		
AWS CloudTrail Data	2023 年 6 月 16 日		
Amazon CloudWatch Internet Monitor	2023 年 6 月 16 日		
Amazon Interacti ve Video Service RealTime	2023 年 6 月 16 日		
AWS IoT TwinMaker	2023 年 6 月 16 日		
Amazon OpenSearch Ingestion	2023 年 6 月 16 日		
AWS Telco Network Builder	2023 年 6 月 16 日		
Amazon VPC Lattice	2023 年 6 月 16 日		
AWS Backup Storage	2023 年 2 月 17 日		
Amazon Chime Media Pipelines	2023 年 2 月 17 日		
Amazon Chime Voice	2023 年 2 月 17 日		
AWS Clean Rooms	2023 年 2 月 17 日	2024年1月18日	
Amazon CodeCatalyst	2023 年 2 月 17 日		
Amazon Connect Cases	2023 年 2 月 17 日		
AWS Control Tower	2023 年 2 月 17 日		

服務	初步支援	Updated	
Amazon DocumentDB Elastic Clusters	2023 年 2 月 17 日		
Amazon EMR Serverless	2023 年 2 月 17 日		
Amazon IVS Chat	2023 年 2 月 17 日		
Amazon Kendra Intelligent Ranking	2023 年 2 月 17 日		
AWS HealthOmics	2023 年 2 月 17 日		
Amazon Redshift Serverless	2023 年 2 月 17 日		
Amazon Security Lake	2023 年 2 月 17 日		
AWS Health	2023 年 2 月 17 日		
AWS IoT FleetWise	2023 年 2 月 17 日		
AWS IoT RoboRunner	2023 年 2 月 17 日		
AWS Mainframe Modernization	2023 年 2 月 17 日		
AWS Migration Hub Orchestrator	2023 年 2 月 17 日		
AWS Private 5G	2023 年 2 月 17 日		
AWS 資源總管	2023 年 2 月 17 日		
AWS SimSpace Weaver	2023 年 2 月 17 日		
AWS Support App	2023 年 2 月 17 日		

服務	初步支援	Updated	
CloudWatch Observability Access Manager	2023 年 2 月 17 日		
EventBridge Pipes	2023 年 2 月 17 日		
EventBridge Scheduler	2023 年 2 月 17 日		
IAM Roles Anywhere	2023 年 2 月 17 日		
Kinesis Video WebRTC Storage	2023 年 2 月 17 日		
License Manager Linux Subscriptions	2023 年 2 月 17 日		
License Manager User Subscriptions	2023 年 2 月 17 日		
OpenSearch Serverless	2023 年 2 月 17 日		
Route 53 ARC Zonal Shift	2023 年 2 月 17 日		
SageMaker Geospatial	2023 年 2 月 17 日		
SageMaker Metrics	2023 年 2 月 17 日		
Systems Manager for SAP	2023 年 2 月 17 日		
AWS Account Management	2022 年 4 月 19 日		
AWS Amplify	2021 年 9 月 30 日		

服務	初步支援	Updated	
AWS App Mesh	2021 年 9 月 30 日		
AWS App Runner	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS AppConfig	2021 年 9 月 30 日		
AWS AppConfig Data	2022 年 4 月 19 日		
AWS AppSync	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Application Discovery Service	2021 年 9 月 30 日		
AWS Application Migration Service	2021 年 9 月 30 日		
AWS Audit Manager	2021 年 9 月 30 日		
AWS Auto Scaling Plans	2021 年 9 月 30 日		
AWS Backup	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Backup gateway	2022 年 4 月 19 日	2023 年 2 月 17 日	
AWS Batch	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Billing Conductor	2022 年 7 月 26 日	2023 年 2 月 17 日	
AWS Budgets	2021 年 9 月 30 日		
AWS Certificate Manager	2021 年 9 月 30 日		
AWS Private Certificate Authority	2021 年 9 月 30 日		
AWS Cloud Map	2021 年 9 月 30 日		

服務	初步支援	Updated	
AWS Cloud9	2021 年 9 月 30 日		
AWS CloudFormation	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS CloudHSM	2021 年 9 月 30 日		
AWS CloudHSM	2021 年 9 月 30 日		
AWS CloudTrail	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Cloud Control	2022 年 4 月 19 日		
AWS CodeBuild	2021 年 9 月 30 日		
AWS CodeCommit	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS CodeDeploy	2021 年 9 月 30 日		
AWS CodePipeline	2021 年 9 月 30 日		
AWS CodeStar	2021 年 9 月 30 日		
AWS CodeStar	2021 年 9 月 30 日		
AWS CodeStar	2021 年 9 月 30 日		
AWS Compute Optimizer	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Config	2021 年 9 月 30 日	2022 年 7 月 26 日	
AWS Cost Explorer Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Cost and Usage Report	2021 年 9 月 30 日		
AWS Data Exchange	2021 年 9 月 30 日	2022 年 7 月 26 日	
AWS Data Pipeline	2021 年 9 月 30 日		

服務	初步支援	Updated	
AWS DataSync	2021 年 9 月 30 日	2022 年 7 月 26 日	
AWS Database Migration Service	2021 年 9 月 30 日		
AWS Device Farm	2021 年 9 月 30 日		
AWS Direct Connect	2021 年 9 月 30 日		
AWS Directory Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS EC2 Instance Connect	2021 年 9 月 30 日		
AWS Elastic Beanstalk	2021 年 9 月 30 日		
AWS Elemental MediaLive	2021 年 9 月 30 日		
AWS Elemental MediaPackage	2021 年 9 月 30 日		
AWS Elemental MediaPackage VOD	2021 年 9 月 30 日		
AWS Elemental MediaStore	2021 年 9 月 30 日		
AWS Fault Injection Service	2021 年 9 月 30 日		
AWS Firewall Manager	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Glue	2021 年 9 月 30 日	2023 年 2 月 17 日	

服務	初步支援	Updated	
AWS Glue DataBrew	2021 年 9 月 30 日		
AWS IoT Greengrass	2021 年 9 月 30 日		
AWS Ground Station	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Identity and Access Management	2021 年 9 月 30 日		
AWS IoT	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS IoT 1-Click	2021 年 9 月 30 日		
AWS IoT Analytics	2021 年 9 月 30 日		
AWS IoT Core Device Advisor	2021 年 9 月 30 日		
AWS IoT Events	2021 年 9 月 30 日		
AWS IoT Events 資料	2021 年 9 月 30 日		
AWS IoT Fleet Hub	2021 年 9 月 30 日		
AWS IoT Greengrass Version 2	2021 年 9 月 30 日		
AWS IoT Jobs Data Plane	2021 年 9 月 30 日		
AWS IoT Secure Tunneling	2021 年 9 月 30 日		
AWS IoT SiteWise	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS IoT Things Graph	2021 年 9 月 30 日		
AWS IoT Wireless	2021 年 9 月 30 日	2023 年 2 月 17 日	

服務	初步支援	Updated	
AWS Key Management Service	2021 年 9 月 30 日	2022 年 7 月 26 日	
AWS Lake Formation	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Lambda	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS License Manager	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Marketplace	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Marketplace Commerce Analytics	2021 年 9 月 30 日		
AWS Marketplace Entitlement Service	2021 年 9 月 30 日		
AWS Elemental MediaTailor	2021 年 9 月 30 日	2022 年 7 月 26 日	
AWS Migration Hub	2021 年 9 月 30 日		
AWS Migration Hub Config	2021 年 9 月 30 日		
AWS Migration Hub 策略建議	2022 年 4 月 19 日	2023 年 2 月 17 日	
AWS Mobile	2021 年 9 月 30 日		
AWS Network Firewall	2021 年 9 月 30 日		
AWS OpsWorks	2021 年 9 月 30 日		
AWS OpsWorks CM	2021 年 9 月 30 日		
AWS Organizations	2021 年 9 月 30 日	2023 年 2 月 17 日	

服務	初步支援	Updated	
AWS Outposts	2021 年 9 月 30 日		
AWS Panorama	2022 年 4 月 19 日	2023 年 2 月 17 日	
Amazon Relationa l Database Service Performance Insights	2021 年 9 月 30 日		
AWS 價格表	2021 年 9 月 30 日		
Amazon Relational Database Service	2021 年 9 月 30 日		
AWS Resilience Hub	2022 年 4 月 19 日		
AWS Resource Access Manager	2021 年 9 月 30 日		
AWS Resource Groups	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Resource Groups Tagging API	2021 年 9 月 30 日		
AWS RoboMaker	2021 年 9 月 30 日		
AWS IAM Identity Center	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS SSO OIDC	2021 年 9 月 30 日		
AWS Secrets Manager	2021 年 9 月 30 日		
AWS Security Token Service	2021 年 9 月 30 日		
AWS Security Hub	2021 年 9 月 30 日		

服務	初步支援	Updated	
AWS Server Migration Service	2021 年 9 月 30 日		
AWS Service Catalog	2021 年 9 月 30 日		
AWS Service Catalog AppRegistry	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Shield	2021 年 9 月 30 日		
AWS Signer	2021 年 9 月 30 日		
AWS IAM Identity Center	2021 年 9 月 30 日		
AWS IAM Identity Center Admin	2021 年 9 月 30 日		
AWS Step Functions	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS Storage Gateway	2021 年 9 月 30 日		
AWS Support	2021 年 9 月 30 日		
AWS Transfer Family	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS WAF	2021 年 9 月 30 日		
AWS WAF Regional	2021 年 9 月 30 日		
AWS WAFV2	2021 年 9 月 30 日		
AWS Well-Architected Tool	2021 年 9 月 30 日	2023 年 2 月 17 日	
AWS X-Ray	2021 年 9 月 30 日	2023 年 2 月 17 日	

服務	初步支援	Updated	
AWS Marketplace Metering Service	2021 年 9 月 30 日		
AWS Serverless Application Repository	2021 年 9 月 30 日		
AWS Identity and Access Management Access Analyzer	2021 年 9 月 30 日		
Alexa for Business	2021 年 9 月 30 日		
Amazon API Gateway	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon API Gateway	2021 年 9 月 30 日		
Amazon AppIntegrations	2021 年 9 月 30 日		
Amazon AppStream 2.0	2021 年 9 月 30 日		
Amazon AppFlow	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Athena	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Augmented AI	2021 年 9 月 30 日		
Amazon Braket	2021 年 9 月 30 日		
Amazon Chime	2021 年 9 月 30 日		
Amazon Chime Meetings	2022 年 4 月 19 日	2023 年 2 月 17 日	
Amazon Cloud Directory	2021 年 9 月 30 日		

服務	初步支援	Updated	
Amazon CloudFront	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon CloudSearch	2021 年 9 月 30 日		
Amazon CloudWatch	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon CloudWatch Application Insights	2021 年 9 月 30 日		
CloudWatch Evidently	2022 年 4 月 19 日		
Amazon CloudWatch Logs	2021 年 9 月 30 日		
Amazon CloudWatch RUM	2022 年 4 月 19 日	2023 年 2 月 17 日	
Amazon CloudWatch Synthetics	2021 年 9 月 30 日		
Amazon CodeGuru Profiler	2021 年 9 月 30 日		
Amazon CodeGuru Reviewer	2021 年 9 月 30 日		
Amazon Cognito	2021 年 9 月 30 日		
Amazon Cognito Identity Provider	2021 年 9 月 30 日		
Amazon Cognito Sync	2021 年 9 月 30 日		
Amazon Comprehend	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Comprehend Medical	2021 年 9 月 30 日		

服務	初步支援	Updated	
Amazon Connect Contact Lens	2021 年 9 月 30 日		
Amazon Connect Participant Service	2021 年 9 月 30 日		
Amazon Connect	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Connect Voice ID	2022 年 4 月 19 日		
Amazon Connect Wisdom	2022 年 4 月 19 日		
Amazon Data Lifecycle Manager	2021 年 9 月 30 日		
Amazon Detective	2021 年 9 月 30 日		
Amazon DevOps Guru	2021 年 9 月 30 日	2022 年 7 月 26 日	
Amazon DocumentDB (with MongoDB compatibility)	2021 年 9 月 30 日		
Amazon DynamoDB	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon DynamoDB Streams	2021 年 9 月 30 日		
Amazon EC2 Container Registry	2021 年 9 月 30 日		
Amazon EC2 Container Service	2021 年 9 月 30 日	2023 年 2 月 17 日	

服務	初步支援	Updated	
Amazon EC2 Systems Manager	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon EMR	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon ElastiCache	2021 年 9 月 30 日		
Amazon Elastic Inference	2021 年 9 月 30 日		
Amazon Elastic Block Store	2021 年 9 月 30 日		
Amazon Elastic Compute Cloud	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Elastic Container Registry Public	2021 年 9 月 30 日		
Amazon Elastic File System	2021 年 9 月 30 日		
Amazon Elastic Kubernetes Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon EMR	2021 年 9 月 30 日		
Amazon Elastic Transcoder	2021 年 9 月 30 日		
Amazon OpenSearch Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon OpenSearch Service	2022 年 4 月 19 日	2023 年 2 月 17 日	
Amazon EventBridge	2021 年 9 月 30 日	2023 年 2 月 17 日	

服務	初步支援	Updated	
Amazon FSx	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Forecast Query	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Forecast Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Fraud Detector	2021 年 9 月 30 日		
Amazon GameLift	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon GameSparks	2022 年 7 月 26 日		
Amazon S3 Glacier	2021 年 9 月 30 日		
Amazon GuardDuty	2021 年 9 月 30 日		
AWS HealthLake	2021 年 9 月 30 日		
Amazon Honeycode	2021 年 9 月 30 日		
Amazon Inspector	2021 年 9 月 30 日		
Amazon Inspector V2	2022 年 4 月 19 日		
Amazon Interactive Video Service	2021 年 9 月 30 日		
Amazon Kendra	2021 年 9 月 30 日		
Amazon Kinesis	2021 年 9 月 30 日		
Amazon Kinesis Analytics	2021 年 9 月 30 日		

服務	初步支援	Updated	
Amazon Kinesis Analytics V2	2021 年 9 月 30 日		
Amazon Kinesis Firehose	2021 年 9 月 30 日		
Amazon Kinesis Video Signaling Channels	2021 年 9 月 30 日		
Amazon Kinesis Video Streams	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Kinesis Video Streams Archived Media	2021 年 9 月 30 日		
Amazon Kinesis video stream	2021 年 9 月 30 日		
Amazon Lex Model Building Service	2021 年 9 月 30 日		
Amazon Lex Model Building Service V2	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Lex	2021 年 9 月 30 日		
Amazon Lex Runtime V2	2021 年 9 月 30 日		
Amazon Lightsail	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Location Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Lookout for Equipment	2021 年 9 月 30 日		

服務	初步支援	Updated	
Amazon Lookout for Metrics	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Lookout for Vision	2021 年 9 月 30 日		
Amazon MQ	2021 年 9 月 30 日		
Amazon Macie	2021 年 9 月 30 日		
Amazon Macie 2	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Managed Blockchain	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Managed Grafana	2022 年 4 月 19 日	2023 年 2 月 17 日	
Amazon Managed Service for Prometheus	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Managed Streaming for Apache Kafka	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Managed Streaming for Apache Kafkax	2022 年 4 月 19 日		
Amazon Managed Workflows for Apache Airflow	2021 年 9 月 30 日		
Amazon Mechanical Turk	2021 年 9 月 30 日		

服務	初步支援	Updated	
Amazon MemoryDB for Redis	2022 年 4 月 19 日	2023 年 2 月 17 日	
Amazon Nimble Studio	2021 年 9 月 30 日		
Amazon Personalize	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Personalize Events	2021 年 9 月 30 日		
Amazon Personalize Runtime	2021 年 9 月 30 日		
Amazon Pinpoint	2021 年 9 月 30 日		
Amazon Pinpoint Email Service	2021 年 9 月 30 日		
Amazon Pinpoint SMS and Voice Service	2021 年 9 月 30 日		
Amazon Pinpoint SMS and Voice V2 Service	2022 年 7 月 26 日		
Amazon Polly	2021 年 9 月 30 日		
Amazon QLDB	2021 年 9 月 30 日		
Amazon QLDB Session	2021 年 9 月 30 日		
Amazon QuickSight	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Redshift	2021 年 9 月 30 日		

服務	初步支援	Updated	
Amazon Redshift Data API	2021 年 9 月 30 日		
Amazon Rekognition	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Relational Database Service	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Route 53	2021 年 9 月 30 日		
Amazon Route 53 Recovery Control Config	2021 年 9 月 30 日	2022 年 7 月 26 日	
Amazon Route 53 Domains	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon Route 53 Resolver	2021 年 9 月 30 日		
Amazon S3 on Outposts	2021 年 9 月 30 日	2022 年 7 月 26 日	
Amazon SageMaker Runtime Feature Store Runtime	2021 年 9 月 30 日		
Amazon SageMaker Runtime Runtime	2021 年 9 月 30 日		
Amazon SageMaker	2021 年 9 月 30 日	2023 年 2 月 17 日	
Amazon SageMaker Edge Manager	2021 年 9 月 30 日		
Amazon Simple Email Service	2021 年 9 月 30 日		

服務	初步支援	Updated
Amazon Simple Email Service V2	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon Simple Notification Service	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon Simple Queue Service	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon Simple Storage Service	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon Simple Workflow Service	2021 年 9 月 30 日	
Amazon Textract	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon Transcribe	2021 年 9 月 30 日	
Amazon Translate	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon WorkDocs	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon WorkMail	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon WorkMail Message Flow	2021 年 9 月 30 日	
Amazon WorkSpaces	2021 年 9 月 30 日	2023 年 2 月 17 日
Amazon WorkSpaces Web	2022 年 4 月 19 日	2023 年 2 月 17 日
Amplify	2021 年 9 月 30 日	
Amplify UI Builder	2022 年 4 月 19 日	2023 年 2 月 17 日
Application Auto Scaling	2021 年 9 月 30 日	

服務	初步支援	Updated	
Amazon EC2 Auto Scaling	2021 年 9 月 30 日	2023 年 2 月 17 日	
CodeArtifact	2021 年 9 月 30 日		
DynamoDB Accelerator	2021 年 9 月 30 日		
EC2 Image Builder	2021 年 9 月 30 日		
AWS Elastic Disaster Recovery	2022 年 4 月 19 日	2023 年 2 月 17 日	
Elastic Load Balancing	2021 年 9 月 30 日		
Elastic Load Balancing V2	2021 年 9 月 30 日		
MediaConnect	2021 年 9 月 30 日		
Amazon S3 Control	2021 年 9 月 30 日	2023 年 2 月 17 日	
Recycle Bin for Amazon EBS	2022 年 4 月 19 日	2023 年 2 月 17 日	
Savings Plans	2021 年 9 月 30 日		
Amazon EventBridge Schema Registry	2021 年 9 月 30 日		
Service Quotas	2021 年 9 月 30 日		
AWS Snowball	2021 年 9 月 30 日		

Step Functions 的範例專案

在AWS Step Functions主控台中，您可以選擇下列其中一個入門範本，將狀態機器部署到您的AWS 帳戶. 這些初學者範 ready-to-run 本是範例專案，可自動建立工作流程 proptotype 和定義，以及專案的所有相關AWS資源。

您可以使用這些範例專案依原樣部署和執行它們，或使用工作流程原型來建置它們。如果您建置在這些專案之上，Step Functions 會建立工作流程原型，但不會部署工作流程定義中列出的資源。

當您部署範例專案時，它們會佈建功能完整的狀態機器，並建立要執行的狀態機器的相關資源。當您建立範例專案時，Step Functions 會用AWS CloudFormation來建立狀態機器所參考的相關資源。

主題

- 管理批次工作 (AWS Batch、Amazon SNS)
- 管理容器工作 (Amazon ECS、Amazon SNS)
- 傳輸資料記錄 (Lambda、DynamoDB、Amazon SQS)
- 投票 Job 狀態 (Lambda、 AWS Batch)
- 任務計時器（ Lambda，Amazon SNS ）
- 回呼模式範例 (Amazon SQS、Amazon SNS、Lambda)
- 管理 Amazon EMR Job
- 執行EMR Serverless工作
- 在工作流程中啟動工作流程 (Step Functions、Lambda)
- 使用「地圖」狀態動態處理資料
- 使用分散式地圖處理 CSV 檔案
- 使用分散式地圖處理 Amazon S3 儲存貯體中的資料
- 訓練機器學習模型
- 調校機器學習模型
- 處理來自 Amazon SQS 的大量訊息 (快速工作流程)
- 選擇性檢查點範例 (快速工作流程)
- 建立 AWS CodeBuild 專案 (CodeBuildAmazon SNS)
- 預先處理資料並訓練機器學習模型

- Lambda 協調流程範例
- 開始 Athena 查詢
- 執行多個查詢（Amazon Athena，Amazon SNS）
- 查詢大型資料集 (Amazon Athena、Amazon S3 AWS Glue、Amazon SNS)
- 讓資料保持最新狀態 (Amazon Athena、Amazon S3 AWS Glue)
- 管理 Amazon EKS 叢集
- 呼叫 API Gateway
- 使用 API Gateway 整合呼叫在 Fargate 上執行的微服務
- 將自訂事件傳送至 EventBridge
- 叫用同步快速工作流
- 使用亞馬遜紅移 (Lambda、亞馬 Amazon Redshift 資料 API) 執行 ETL/ELT 工作流程
- 使用Step Functions和AWS Batch錯誤處理
- 扇出一 AWS Batch 份工作
- AWS Batch 與 Lambda
- 使用執行 AI 提示鏈結 Amazon Bedrock

管理批次工作 (AWS Batch、Amazon SNS)

此範例專案示範如何提交 AWS Batch 任務，然後根據任務成功或失敗傳送 Amazon SNS 通知。部署此範例專案將建立 AWS Step Functions 狀態機器、AWS Batch 任務和 Amazon SNS 主題。

在此專案中，Step Functions 會使用狀態機器同步呼叫 AWS Batch 任務。它會等待任務成功或失敗，然後傳送 Amazon SNS 主題，其中包含任務成功或失敗的訊息。

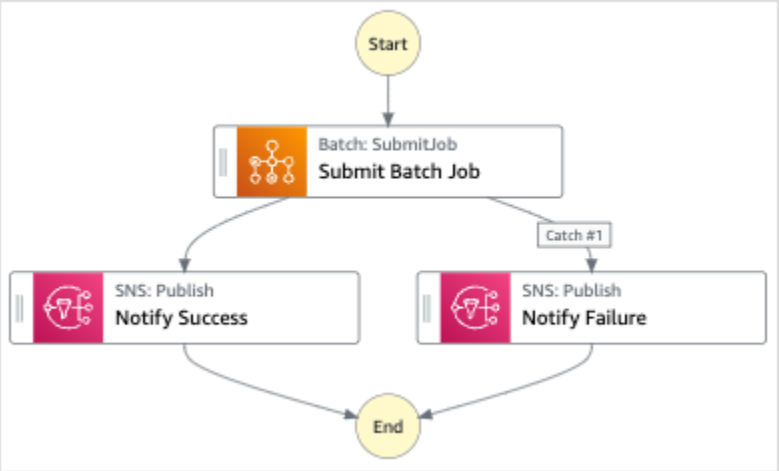
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Manage a batch job`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇「管理批次工作」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

此範例專案會部署下列資源：

- 一 AWS Batch 份工作
- Amazon SNS 主題
- 一个 AWS Step Functions 状态机
- 相關的 AWS Identity and Access Management (IAM) 角色

下圖顯示「管理批次工作範例專案」的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 AWS Batch 和 Amazon SNS 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN)，並傳遞至服務 API `Parameters` 來瞭解步驟函數如何控制 AWS Batch 和 Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch
 job completion",
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
BatchJobQueue-7049d367474b4dd",
        "JobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/
BatchJobDefinition-74d55ec34c4643c:1"
      },
      "Next": "Notify Success",
      "Catch": [
```

```
          {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "Notify Failure"
          }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions succeeded",
        "TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-
SNSTopic-1J757CVBQ2KHM"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions failed",
        "TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-
SNSTopic-1J757CVBQ2KHM"
      },
      "End": true
    }
  }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:sns:ap-northeast-1:123456789012:ManageBatchJob-SNSTopic-
JHLYYG7AZPZI"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:ap-northeast-1:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

管理容器工作 (Amazon ECS、Amazon SNS)

此範例專案示範如何執行工AWS Fargate作，然後根據該工作是成功還是失敗傳送Amazon SNS通知。部署此範例專案將會建立AWS Step Functions狀態機器、Fargate叢集和Amazon SNS主題。

在這個項目中，Step Functions使用狀態機同步調用Fargate任務。它會等待任務成功或失敗，然後傳送 Amazon SNS 主題，其中包含工作成功或失敗的訊息。

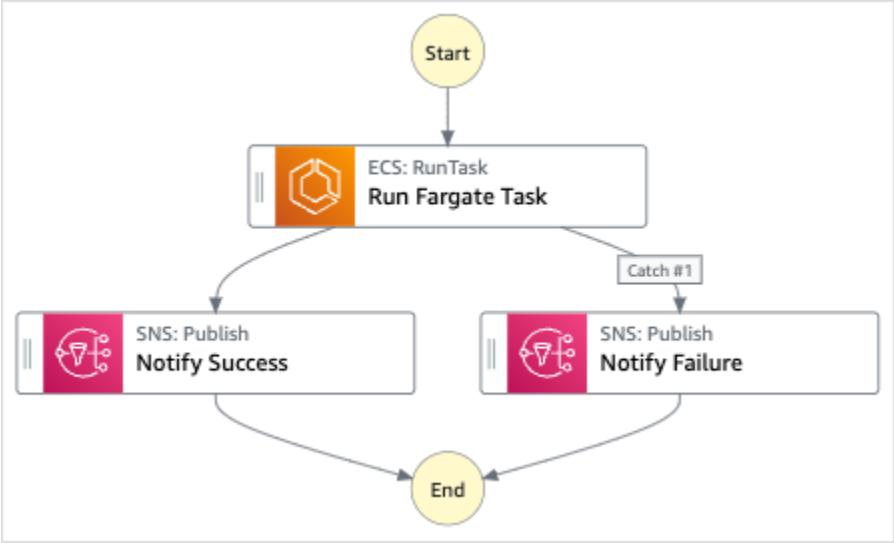
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **Manage a container task**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [管理容器工作]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - AWS Fargate 叢集
   - Amazon SNS 主題
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下圖顯示「管理容器」工作範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. 「Step Functions」主控台會將您引導至標題為您的執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 AWS Fargate 和 Amazon SNS 整合。瀏覽此範例狀態機器，瞭解 Step Functions 如何使用狀態機器同步呼叫 Fargate 任務、等待工作成功或失敗，以及傳送 Amazon SNS 主題，其中包含工作是成功還是失敗的訊息。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS
 Fargate task completion",
  "StartAt": "Run Fargate Task",
  "TimeoutSeconds": 3600,
  "States": {
    "Run Fargate Task": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:states:::ecs:runTask.sync",
      "Parameters": {
        "LaunchType": "FARGATE",
        "Cluster": "arn:aws:ecs:ap-northeast-1:123456789012:cluster/
FargateTaskNotification-ECSCluster-VHLR20IF9IMP",
        "TaskDefinition": "arn:aws:ecs:ap-northeast-1:123456789012:task-definition/
FargateTaskNotification-ECSTaskDefinition-13YOJT8Z2LY5Q:1",
        "NetworkConfiguration": {
          "AwsvpcConfiguration": {
            "Subnets": [
              "subnet-07e1ad3abcfce6758",
              "subnet-04782e7f34ae3efdb"
            ],
            "AssignPublicIp": "ENABLED"
          }
        }
      },
      "Next": "Notify Success",
      "Catch": [
          {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "Notify Failure"
          }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "AWS Fargate Task started by Step Functions succeeded",
        "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "AWS Fargate Task started by Step Functions failed",
        "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
      },
      "End": true
```

```
    }
  }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。最佳做法是僅包含 IAM 政策中必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:ap-northeast-1:123456789012:task-definition/
FargateTaskNotification-ECSTaskDefinition-13YOJT8Z2LY5Q:1"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecs:StopTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
```

```
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:ap-northeast-1:123456789012:rule/
StepFunctionsGetEventsForECSTaskRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

傳輸資料記錄 (Lambda、DynamoDB、Amazon SQS)

此範例專案示範如何反覆讀取Amazon DynamoDB資料表中的項目，並使用Step Functions狀態機將這些項目傳送至Amazon SQS佇列。部署此範例專案將會建立Step Functions狀態機器、DynamoDB資料表、AWS Lambda函數和Amazon SQS佇列。

在這個專案中，Step Functions使用Lambda函數來填入資DynamoDB料表。狀態機器也會使用`for`迴圈來讀取每個項目，然後將每個項目傳送至Amazon SQS佇列。

步驟 1：建立狀態機器並佈建資源

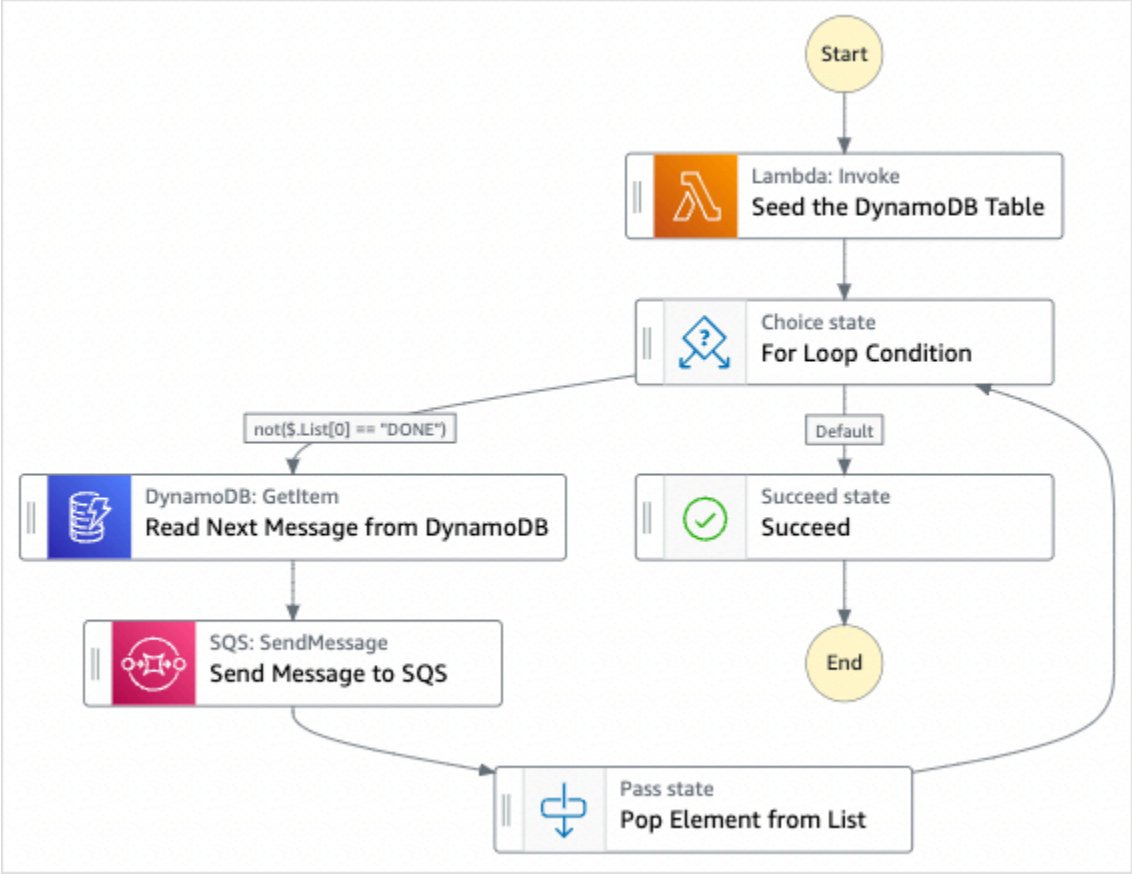
1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Transfer data records`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [傳送資料記錄]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 用於植入 DynamoDB 資料表的 Lambda 函數
   - Amazon SQS 隊列
   - DynamoDB 資料表

- 一个 AWS Step Functions 状态机
- 相關的 AWS Identity and Access Management (IAM) 角色

下列影像展示了「轉移資料記錄」範例專案的工作流程圖表：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. 「Step Functions」主控台會將您引導至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 DynamoDB 和 Amazon SQS 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN) 並傳遞至服務 API `Parameters` 來控制 DynamoDB 和 Amazon SQS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment" : "An example of the Amazon States Language for reading messages from a
 DynamoDB table and sending them to SQS",
  "StartAt": "Seed the DynamoDB Table",
  "TimeoutSeconds": 3600,
  "States": {
    "Seed the DynamoDB Table": {
```

```
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:sqsconnector-
SeedingFunction-T3U43VYDU5OQ",
      "ResultPath": "$.List",
      "Next": "For Loop Condition"
    },
    "For Loop Condition": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.List[0]",
            "StringEquals": "DONE"
          },
          "Next": "Read Next Message from DynamoDB"
        }
      ],
      "Default": "Succeed"
    },
    "Read Next Message from DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "sqsconnector-DDBTable-1CAFOJWP8QD6I",
        "Key": {
          "MessageId": {"S.$": "$.List[0]"}
        }
      },
      "ResultPath": "$.DynamoDB",
      "Next": "Send Message to SQS"
    },
    "Send Message to SQS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sqs:sendMessage",
      "Parameters": {
        "MessageBody.$": "$.DynamoDB.Item.Message.S",
        "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/sqsconnector-
SQSQueue-QVGQBW134PWK"
      },
      "ResultPath": "$.SQS",
      "Next": "Pop Element from List"
    },
    "Pop Element from List": {
      "Type": "Pass",
```

```
      "Parameters": {
        "List.$": "$.List[1:]"
      },
      "Next": "For Loop Condition"
    },
    "Succeed": {
      "Type": "Succeed"
    }
  }
}
```

如需傳遞參數和管理結果的詳細資訊，請參閱以下內容：

- 將參數傳遞至服務 API
- ResultPath

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。最佳做法是僅包含 IAM 政策中必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dynamodb:GetItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:ap-northeast-1:123456789012:table/
TransferDataRecords-DDBTable-3I41R5L5EAGT"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:ap-northeast-1:123456789012:TransferDataRecords-SQSQueue-
BKWXTS09LIW1"
            ],
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:invokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:ap-
northeast-1:123456789012:function:TransferDataRecords-SeedingFunction-VN4KY2TPAZSR"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

投票 Job 狀態 (Lambda、 AWS Batch)

此範例專案會建立 AWS Batch 工作輪詢工作。它實現了一個 AWS Step Functions 狀態機器，用 AWS Lambda 於創建一個`Wait`狀態循環，檢查一個 AWS Batch 作業。

此範例專案會建立並設定所有資源，以便您的 Step Functions 工作流程提交 AWS Batch 工作，並在順利結束之前等待該工作完成。

> **Note**
>
> 您也可以在不使用 Lambda 函數的情況下實作此模式。如需有關 AWS Batch 直接控制的資訊，請參閱AWS Step Functions 搭配其他服務使用。

此範例專案會建立狀態機器、兩個 Lambda 函數和一個 AWS Batch 佇列，並設定相關的 IAM 許可。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

步驟 1：建立狀態機器並佈建資源

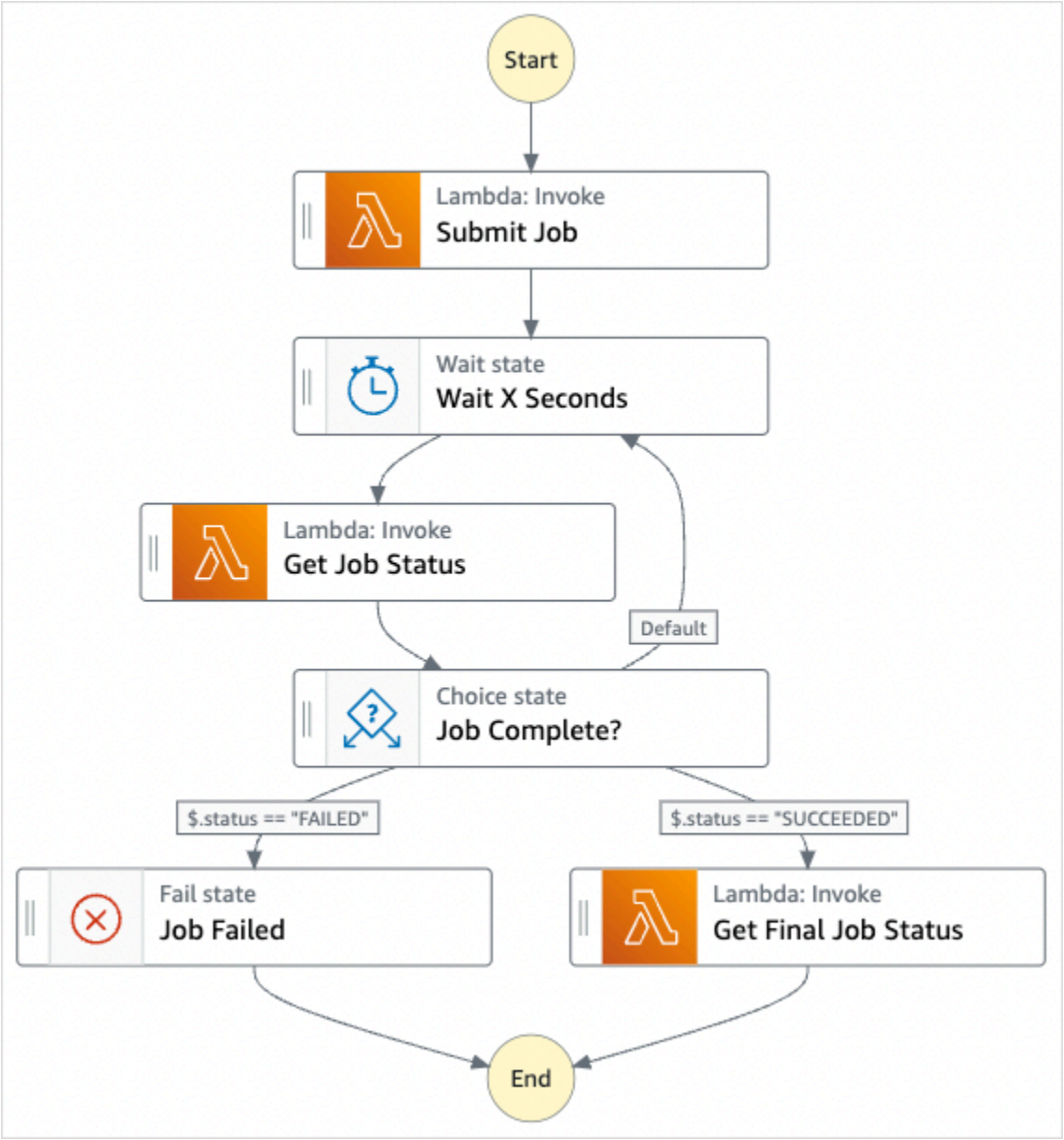
1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。

2. **`Job Poller`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [Job 輪詢器]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 三個 Lambda 函數可提交 AWS Batch 工作、取得已提交 AWS Batch 工作的目前狀態，以及最終工作完成狀態。
   - 一份 AWS Batch 工作
   - 一个 AWS Step Functions 状态机
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示「Job 輪詢者」範例專案的工作流程圖表：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

  > **Tip**
  >
  > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

  準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

  建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

  建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

  > **Important**
  >
  > CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

佈建並部署所有資源之後，會顯示 [開始執行] 對話方塊，其中包含類似下列內容的範例輸入。

```
{
    "jobName": "my-job",
    "jobDefinition": "arn:aws:batch:us-east-2:123456789012:job-definition/
SampleJobDefinition-343f54b445d5312:1",
    "jobQueue": "arn:aws:batch:us-east-2:123456789012:job-queue/
SampleJobQueue-4d9d696031e1449",
    "wait_time": 60
  }
```

> **Note**
>
> `wait_time` 指示每 60 秒迴圈的 `Wait` 狀態。

- 在 [開始執行] 對話方塊中，執行下列動作：

  1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

     > **Note**
     >
     > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

  2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

     如果您選擇執行示範，則不需要提供任何執行輸入。

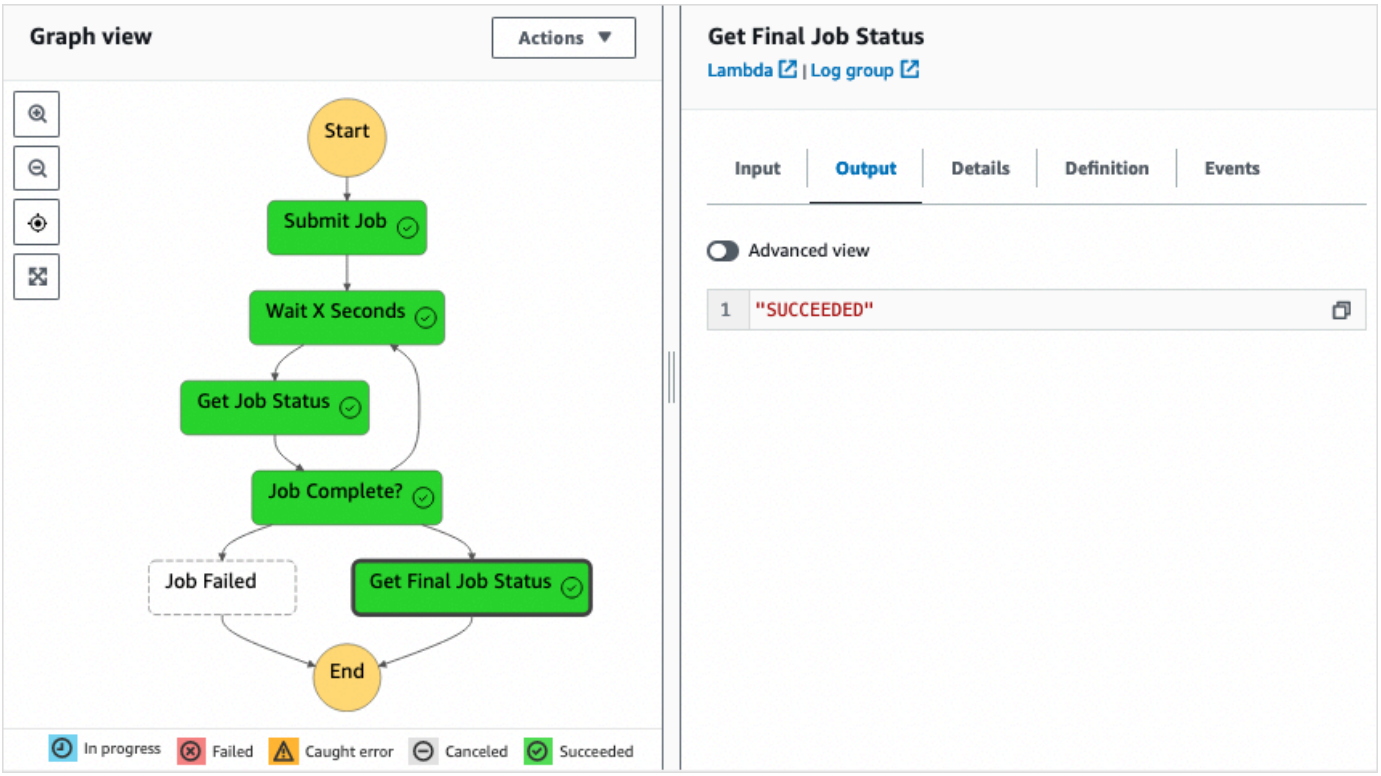
     > **Note**
     >
     > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

  3. 選擇 Start execution (開始執行)。
  4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成後複查執行結果。

     若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

  例如，若要檢視 AWS Batch 工作的狀態變更，以及執行的迴圈結果，請選擇 [輸出] 索引標籤。

  下圖顯示「圖形」檢視中的執行狀態圖形。它也會在「輸出」頁籤中顯示所選步驟的執行輸出。



範例狀態機器程式碼

此範例專案中的狀態機器與整合 AWS Lambda 以提交 AWS Batch 工作。瀏覽此範例狀態機器，以瞭解 Step Functions 如何控制 Lambda 和 AWS Batch.

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "Comment": "An example of the Amazon States Language that runs an AWS Batch job and
 monitors the job until it completes.",
    "StartAt": "Submit Job",
    "States": {
      "Submit Job": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:StepFunctionsSample-JobStatusPol-SubmitJobFunction-
jDaYcl4cx55r",
        "ResultPath": "$.guid",
        "Next": "Wait X Seconds"
      },
      "Wait X Seconds": {
```

```
        "Type": "Wait",
        "SecondsPath": "$.wait_time",
        "Next": "Get Job Status"
      },
      "Get Job Status": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:StepFunctionsSample-JobStatusPoll-
CheckJobFunction-1JkJwY10vonI",
        "Next": "Job Complete?",
        "InputPath": "$.guid",
        "ResultPath": "$.status"
      },
      "Job Complete?": {
        "Type": "Choice",
        "Choices": [
          {
            "Variable": "$.status",
            "StringEquals": "FAILED",
            "Next": "Job Failed"
          },
          {
            "Variable": "$.status",
            "StringEquals": "SUCCEEDED",
            "Next": "Get Final Job Status"
          }
        ],
        "Default": "Wait X Seconds"
      },
      "Job Failed": {
        "Type": "Fail",
        "Cause": "AWS Batch Job Failed",
        "Error": "DescribeJob returned FAILED"
      },
      "Get Final Job Status": {
        "Type": "Task",
        "Resource": "arn:aws::lambda:us-
east-1:111122223333:function:StepFunctionsSample-JobStatusPoll-
CheckJobFunction-1JkJwY10vonI",
        "InputPath": "$.guid",
        "End": true
      }
    }
```

```
}
```

任務計時器（Lambda，Amazon SNS）

此範例專案會建立任務計時器。它實作了實作 AWS Step Functions 狀態的`Wait`狀態機器，並使用傳送 Amazon Simple Notification Service (Amazon SNS) 通知的 AWS Lambda 函數。等候 狀态是等待触发器执行单个工作单元的状态类型。

> **Note**
>
> 此範例專案實 AWS Lambda 作傳送 Amazon Simple Notification Service (Amazon SNS) 通知的函數。您也可以直接從 Amazon 州語言傳送 Amazon SNS 通知。請參閱AWS Step Functions 搭配其他服務使用。

此範例專案會建立狀態機器、Lambda 函數和 Amazon SNS 主題，並設定相關 AWS Identity and Access Management (IAM) 許可。如需使用 Task Timer (任務計時器) 範例專案建立的資源詳細資訊，請參閱以下內容：

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

- AWS CloudFormation 使用者指南
- 《Amazon Simple Notification Service 開發人員指南》https://docs.aws.amazon.com/sns/latest/dg/
- AWS Lambda 開發人員指南
- IAM 入門指南

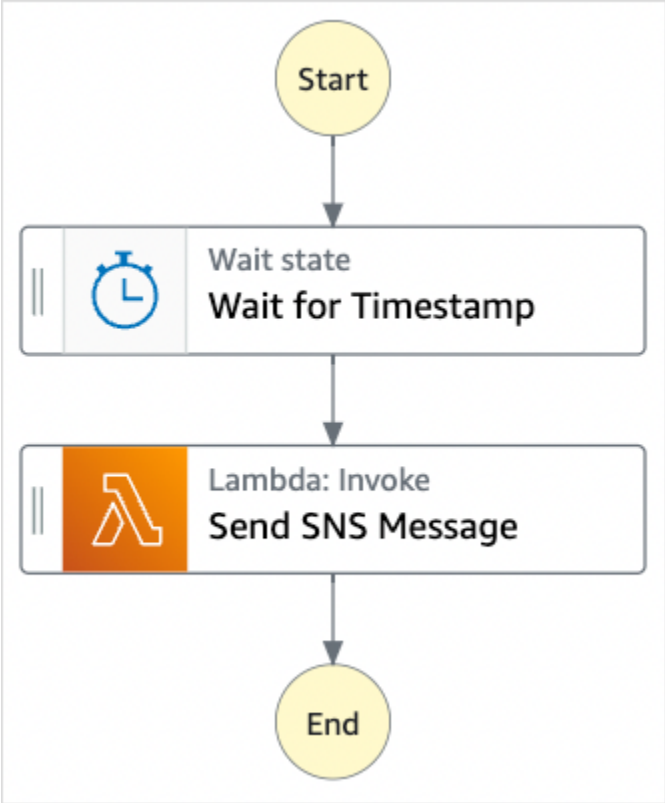
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Task Timer`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [工作計時器]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

此範例專案會部署下列資源：

- 傳送 Amazon SNS 通知的 Lambda 函數。
- 一个 AWS Step Functions 状态机
- 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示工作計時器範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

  > **Tip**
  >
  > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

  準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

  建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

  建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

  > **Important**
  >
  > CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

佈建並部署所有資源之後，會顯示 [開始執行] 對話方塊，其中包含類似下列內容的範例輸入。

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- 在 [開始執行] 對話方塊中，執行下列動作：

1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

   > **Note**
   >
   > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

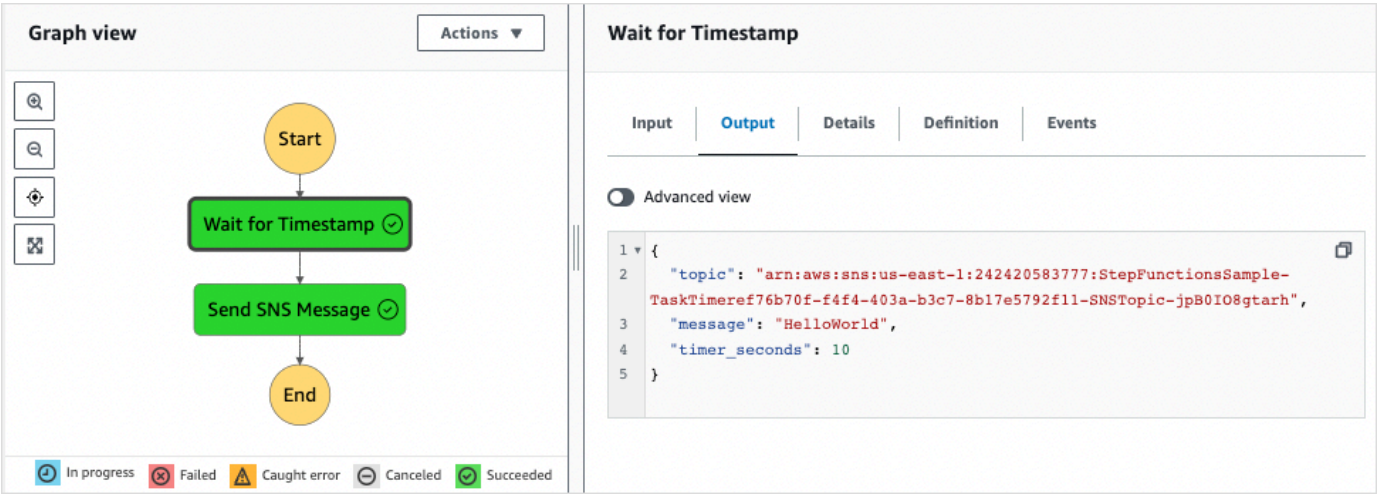
   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

例如，下圖顯示所選步驟的輸出等待時間戳記。此步驟的輸出會作為輸入傳送至「傳送 SNS 訊息」步驟。



回呼模式範例 (Amazon SQS、Amazon SNS、Lambda)

此範例專案示範如何在工作期間 AWS Step Functions 暫停，並等待外部處理序傳回工作開始時產生的工作 Token。

當這個範例專案部署完成且開始執行後，便會發生下列步驟。

1. Step Functions 會將包含任務權杖的訊息傳遞至 Amazon Simple Queue Service (Amazon SQS) 佇列。
2. Step Functions 然後暫停，等待返回該令牌。
3. Amazon SQS 佇列會觸發`SendTaskSuccess`使用相同任務權杖呼叫的 AWS Lambda 函數。
4. 當收到該任務字符時，工作流程就會繼續進行。
5. 此`"Notify Success"`任務會發佈 Amazon Simple Notification Service (Amazon SNS) 訊息，表示已收到回呼。

若要瞭解如何在 Step Functions 式中實作回呼模式，請參閱等候傳回任務字符的回呼。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

步驟 1：建立狀態機器並佈建資源

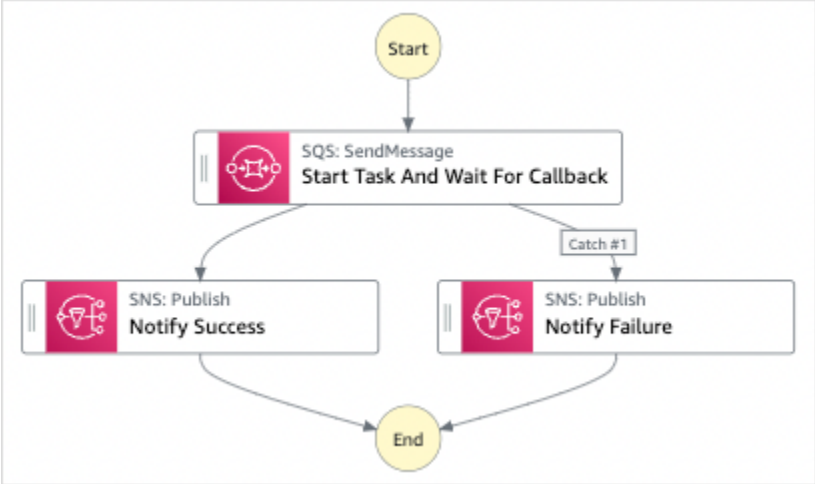
1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Callback pattern example`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇「回呼」模式範例。

3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - Amazon SQS 訊息佇列。
   - 呼叫 Step Functions 函數 API 動作的 Lambda 函數SendTaskSuccess。
   - Amazon SNS 主題，用於通知任務成功或失敗，指出工作流程是否可以繼續。
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下圖顯示回呼模式範例專案範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

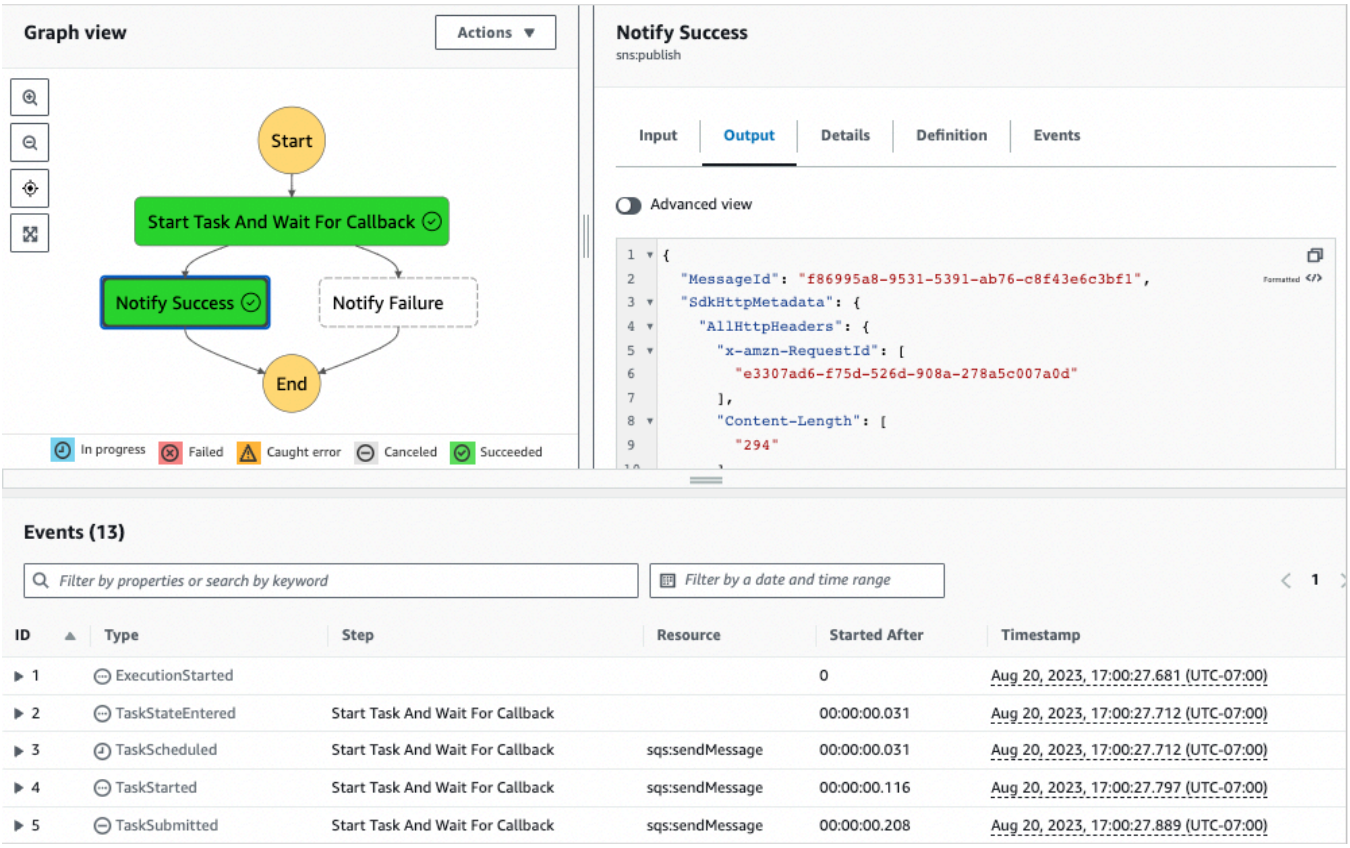
   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. 「Step Functions」主控台會將您引導至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

例如，若要檢閱 Step Functions 如何在工作流程中進行，並從 Amazon SQS 接收回呼，請檢閱「事件」表中的項目。下圖顯示「通知成功」步驟的執行輸出。它還顯示執行事件歷史記錄中的前五個事件。展開每個事件以檢視有關該事件的更多詳細資料。



Lambda 回調示例

若要查看此範例專案的元件如何協同運作，請參閱您 AWS 帳戶中部署的資源。例如，以下是使用工作權杖呼叫 Step Functions 數的 Lambda 函數。

```
console.log('Loading function');
const aws = require('aws-sdk');

exports.lambda_handler = (event, context, callback) => {
    const stepfunctions = new aws.StepFunctions();

    for (const record of event.Records) {
        const messageBody = JSON.parse(record.body);
        const taskToken = messageBody.TaskToken;

        const params = {
            output: "\"Callback task completed successfully.\"",
            taskToken: taskToken
        };
```

```
        console.log(`Calling Step Functions to complete callback task with params
 ${JSON.stringify(params)}`);

        stepfunctions.sendTaskSuccess(params, (err, data) => {
            if (err) {
                console.error(err.message);
                callback(err.message);
                return;
            }
            console.log(data);
            callback(null);
        });
    }
};
```

管理 Amazon EMR Job

這個範例專案示範了 Amazon EMR 和 AWS Step Functions 整合。

它說明如何建立 Amazon EMR 叢集、新增多個步驟並執行它們，然後終止叢集。

> **Important**
>
> Amazon EMR 沒有免費定價方案。執行範例專案將產生費用。您可以在 Amazon EMR 定價頁面上找到定價資訊。Amazon EMR 服務整合的可用性取決於 Amazon EMR API 的可用性。因此，此範例專案在某些 AWS 區域中可能無法正常運作。如需特殊區域的限制，請參閱 Amazon EMR 文件。

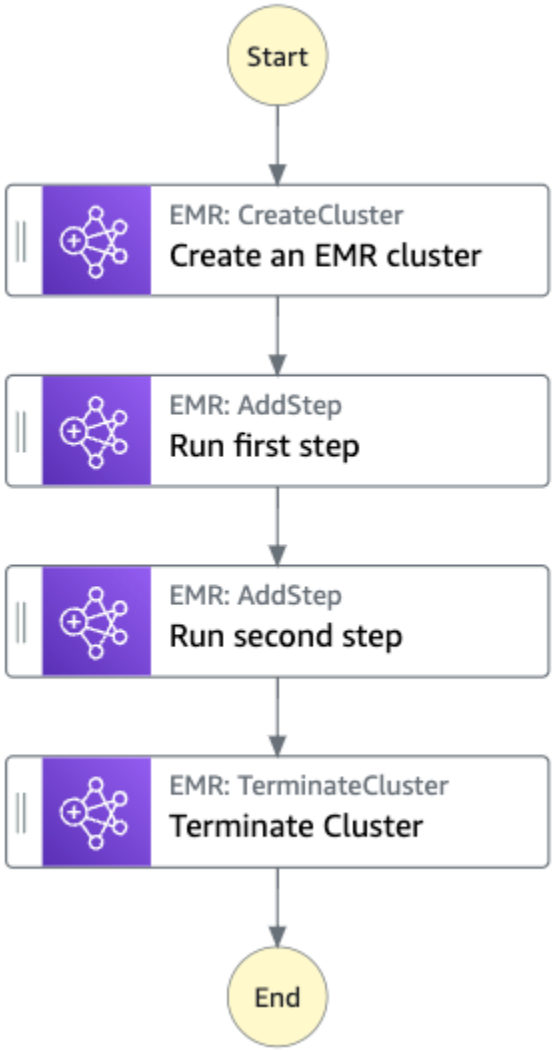
步驟 1：建立狀態機並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Manage an EMR job`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [管理 EMR 工作]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

- Amazon S3 儲存貯體
- Amazon EMR叢集
- 一个 AWS Step Functions 状态机
- 相關的 AWS Identity and Access Management (IAM) 角色

下圖顯示「管理 EMR」工作範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 Amazon EMR 整合。瀏覽此範例狀態機器，瞭解 Step Functions 如何使用狀態機器同步呼叫 Amazon EMR 任務、等待任務成功或失敗，以及終止叢集。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for running jobs on Amazon EMR",
  "StartAt": "Create an EMR cluster",
  "States": {
    "Create an EMR cluster": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:createCluster.sync",
      "Parameters": {
```

```
        "Name": "ExampleCluster",
        "VisibleToAllUsers": true,
        "ReleaseLabel": "emr-5.26.0",
        "Applications": [
          { "Name": "Hive" }
        ],
        "ServiceRole": "<EMR_SERVICE_ROLE>",
        "JobFlowRole": "<EMR_EC2_INSTANCE_PROFILE>",
        "LogUri": "s3://<EMR_LOG_S3_BUCKET>/logs/",
        "Instances": {
          "KeepJobFlowAliveWhenNoSteps": true,
          "InstanceFleets": [
            {
              "Name": "MyMasterFleet",
              "InstanceFleetType": "MASTER",
              "TargetOnDemandCapacity": 1,
              "InstanceTypeConfigs": [
                {
                  "InstanceType": "m5.xlarge"
                }
              ]
            },
            {
              "Name": "MyCoreFleet",
              "InstanceFleetType": "CORE",
              "TargetOnDemandCapacity": 1,
              "InstanceTypeConfigs": [
                {
                  "InstanceType": "m5.xlarge"
                }
              ]
            }
          ]
        }
      },
      "ResultPath": "$.cluster",
      "Next": "Run first step"
    },
    "Run first step": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:addStep.sync",
      "Parameters": {
        "ClusterId.$": "$.cluster.ClusterId",
        "Step": {
```

```
        "Name": "My first EMR step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
          "Jar": "command-runner.jar",
          "Args": ["<COMMAND_ARGUMENTS>"]
        }
      }
    },
    "Retry" : [
      {
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 1,
        "MaxAttempts": 3,
        "BackoffRate": 2.0
      }
    ],
    "ResultPath": "$.firstStep",
    "Next": "Run second step"
  },
  "Run second step": {
    "Type": "Task",
    "Resource": "arn:<PARTITION>:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
      "ClusterId.$": "$.cluster.ClusterId",
      "Step": {
        "Name": "My second EMR step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
          "Jar": "command-runner.jar",
          "Args": ["<COMMAND_ARGUMENTS>"]
        }
      }
    },
    "Retry" : [
      {
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 1,
        "MaxAttempts": 3,
        "BackoffRate": 2.0
      }
    ],
    "ResultPath": "$.secondStep",
    "Next": "Terminate Cluster"
  },
```

```
    "Terminate Cluster": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:terminateCluster",
      "Parameters": {
        "ClusterId.$": "$.cluster.ClusterId"
      },
      "End": true
    }
  }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。最佳做法是僅包含 IAM 政策中必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:RunJobFlow",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:TerminateJobFlows"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::123456789012:role/StepFunctionsSample-EMRJobManagement-
EMRServiceRole-ANPAJ2UCCR6DPCEXAMPLE",
                "arn:aws:iam::123456789012:role/StepFunctionsSample-
EMRJobManagementWJALRXUTNFEMI-ANPAJ2UCCR6DPCEXAMPLE-
EMREc2InstanceProfile-1ANPAJ2UCCR6DPCEXAMPLE"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
```

```
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForEMRRunJobFlowRule"
            ]
        }
    ]
}
```

下列政策可確保 addStep 具有足夠的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:AddJobFlowSteps",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:CancelSteps"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForEMRAddJobFlowStepsRule"
            ]
        }
    ]
}
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

執行EMR Serverless工作

此範例專案示範如何建立和啟動EMR Serverless應用程式。該項目還顯示了如何在該應用程序中運行多個作業。

此範例專案會建立狀態機器、支援 AWS 資源，並設定相關的 IAM 許可。探索此範例專案，瞭解如何使用Step Functions狀態機器執行EMR Serverless作業，或將其用作您自己專案的起點。

> **Important**
>
> EMR Serverless 沒有免費定價方案。執行範例專案將產生費用。您可以在定價頁面上找到 Amazon EMR Serverless 定價資訊。
> 此外，EMR Serverless服務整合的可用性取決於 EMR Serverless API 的可用性。因此，此示例項目可能無法正常工作或在某些項目中可用 AWS 區域。如需EMR Serverless中的可用性的相關資訊，請參閱其他考量主題 AWS 區域。

AWS CloudFormation 範本和其他資源

您可以使用CloudFormation範本來部署此範例專案。此範本會在您的中建立下列資源 AWS 帳戶：

- 狀Step Functions態機。
- 狀態機器的執行角色。此角色授 AWS 服務 與狀態機存取其他資源 (例如EMR ServerlessCreateApplication動作) 所需的權限。
- 的 Job 執行角色EMR Serverless。此角色會授與EMR Serverless工作執行代表您呼叫其他服務時所能承擔的權限。

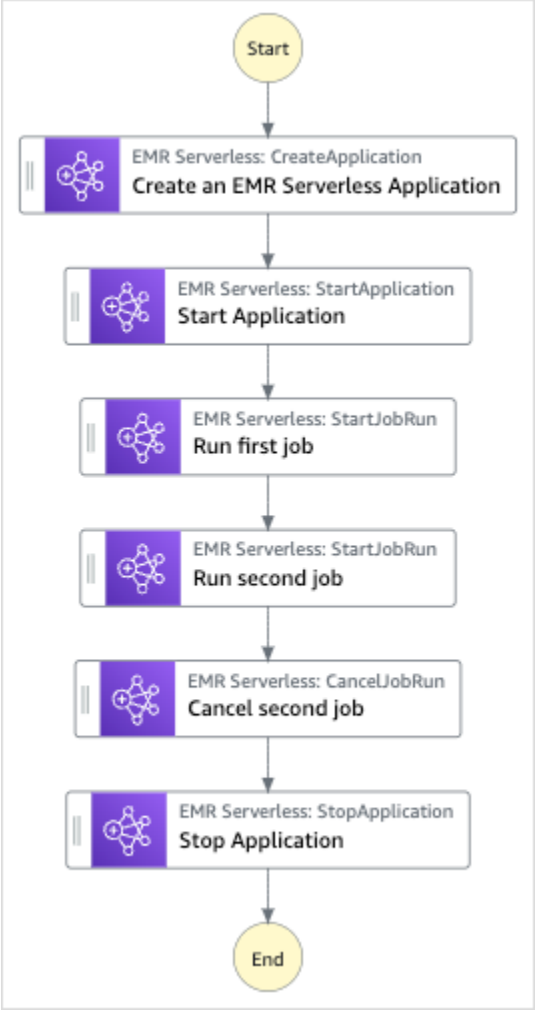
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`EMR Serverless`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [執行EMR Serverless工作]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

此範例專案會部署下列資源：

- Step Functions 狀態機器
- 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示「執行EMR Serverless作業範例專案」的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

- 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

  在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀

態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

在工作流程中啟動工作流程 (Step Functions、Lambda)

此範例專案示範如何使用 AWS Step Functions 狀態機啟動其他狀態機執行。如需有關從另一個狀態機器啟動狀態機器執行的資訊，請參閱從任務狀態開始工作流程執行。

步驟 1：建立狀態機器並佈建資源

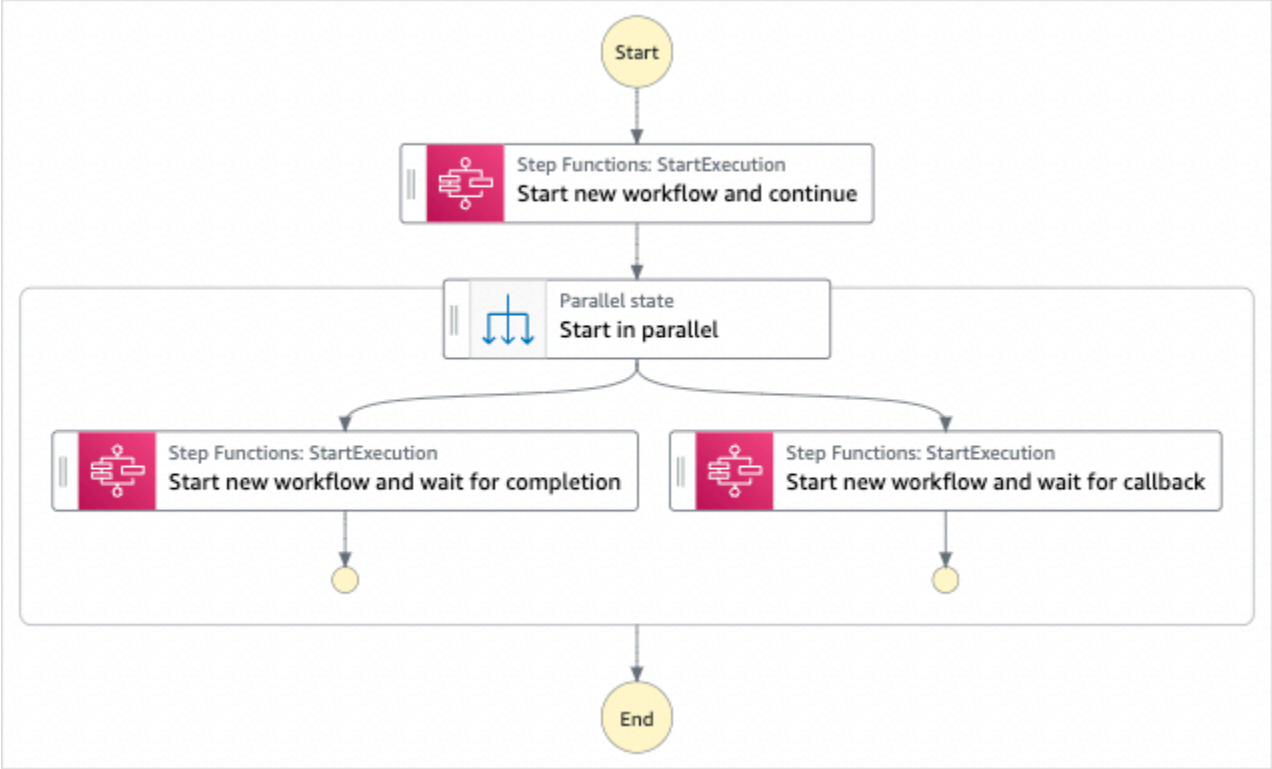
1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 請**`Start a workflow within a workflow`**在搜索框中鍵入內容，然后從返回的搜索結果中選擇「在某個工作流程中啟動某個工作流程」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 一個額外的狀態機。此狀態機器的執行由您執行的狀態機器啟動。
   - 一個回調 Lambda 函數。該函數用於附加的狀態機來實現回調機制。

- 一个 AWS Step Functions 状态机
- 相關的 AWS Identity and Access Management (IAM) 角色

下列影像展示了在工作流程範例專案中啟動工作流程的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

- 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

  在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

  > **Important**
  >
  > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器整合了另一個狀態機器，並 AWS Lambda 將參數直接傳遞給這些資源。

瀏覽此範例狀態機器，以瞭解 Step Functions 如何呼叫其他狀態機器的 `StartExecution` API 動作。它會平行啟動另一部狀態機器的兩個執行個體：一個使用 執行任務 (.sync) 模式，另一個使用 等候傳回任務字符的回呼 模式。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of combining workflows using a Step Functions StartExecution
 task state with various integration patterns.",
  "StartAt": "Start new workflow and continue",
  "States": {
    "Start new workflow and continue": {
      "Comment": "Start an execution of another Step Functions state machine and
 continue",
      "Type": "Task",
      "Resource": "arn:aws:states:::states:startExecution",
      "Parameters": {
        "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:NestingPatternAnotherStateMachine-HZ9gtgspmdun",
        "Input": {
          "NeedCallback": false,
          "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
        }
```

```
      },
      "Next": "Start in parallel"
    },
    "Start in parallel": {
      "Comment": "Start two executions of the same state machine in parallel",
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Start new workflow and wait for completion",
          "States": {
            "Start new workflow and wait for completion": {
              "Comment": "Start an execution of the same
 'NestingPatternAnotherStateMachine' and wait for its completion",
              "Type": "Task",
              "Resource": "arn:aws:states:::states:startExecution.sync",
              "Parameters": {
                "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:NestingPatternAnotherStateMachine-HZ9gtgspmdun",
                "Input": {
                  "NeedCallback": false,
                  "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
                }
              },
              "OutputPath": "$.Output",
              "End": true
            }
          }
        },
        {
          "StartAt": "Start new workflow and wait for callback",
          "States": {
            "Start new workflow and wait for callback": {
              "Comment": "Start an execution and wait for it to call back with a task
 token",
              "Type": "Task",
              "Resource": "arn:aws:states:::states:startExecution.waitForTaskToken",
              "Parameters": {
                "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:NestingPatternAnotherStateMachine-HZ9gtgspmdun",
                "Input": {
                  "NeedCallback": true,
                  "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id",
                  "TaskToken.$": "$$.Task.Token"
```

```
                }
              },
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

使用「地圖」狀態動態處理資料

此範例專案示範使用 Map 狀態的動態平行處理。

在此專案中，Step Functions 數使用 AWS Lambda 函數從 Amazon SQS 佇列中提取訊息，並將這些訊息的 JSON 陣列傳遞至Map狀態。對於佇列中的每個訊息，狀態機器會將訊息寫入 DynamoDB，叫用另一個 Lambda 函數從 Amazon SQS 移除訊息，然後將訊息發佈到 Amazon SNS 主題。

如需Map狀態和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- Map
- AWS Step Functions 搭配其他服務使用

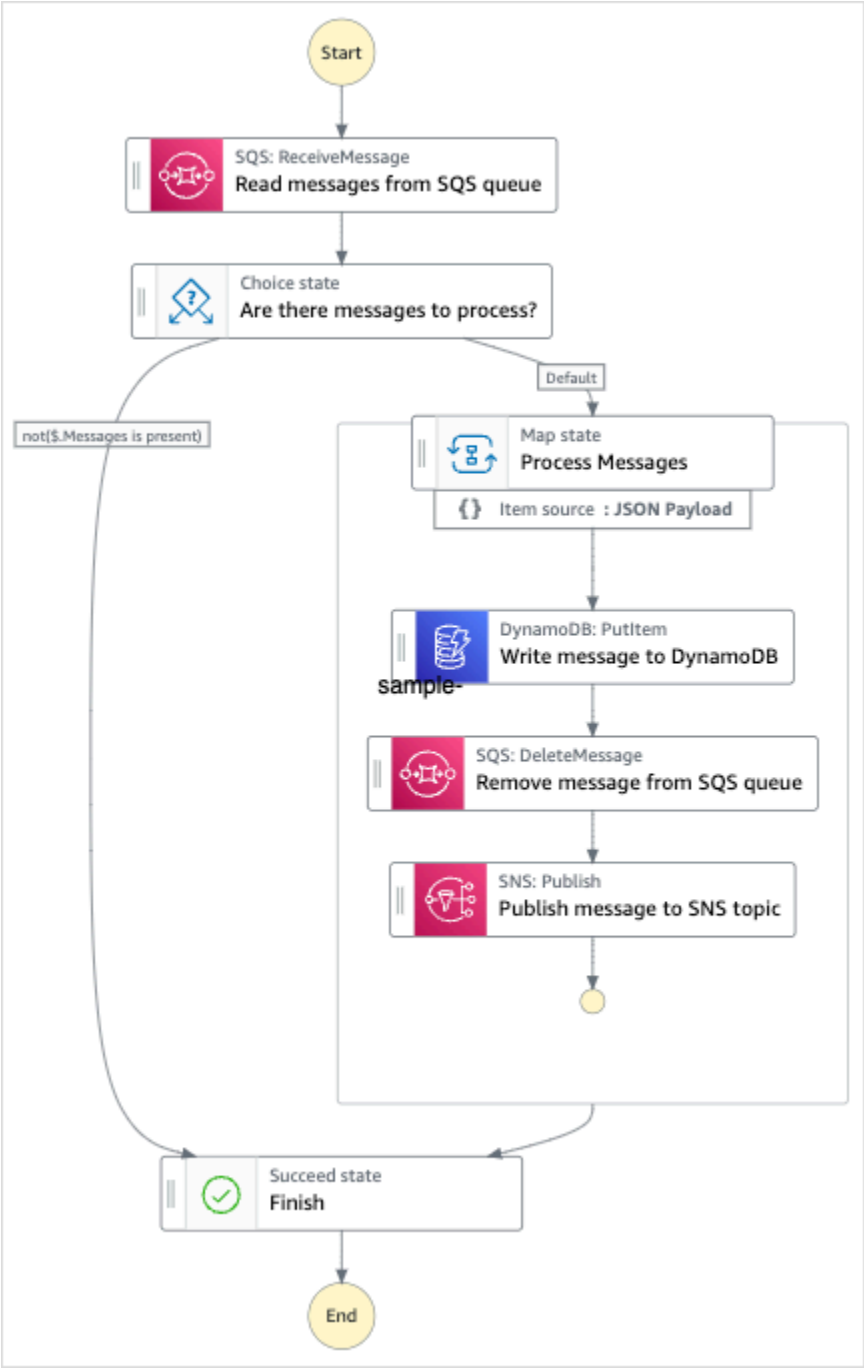
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Dynamically process data with a Map state`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇「動態處理具有對應」狀態的資料。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

- 一種 Amazon SQS 佇列，地圖狀態會反覆讀取和移除訊息。
- 地圖狀態會反覆寫入訊息的 DynamoDB 表格。
- 一個 Amazon SNS 主題，Step Functions 會發佈從 Amazon SQS 佇列讀取的訊息。
- 兩種 AWS Lambda 功能
- 一个 AWS Step Functions 状态机
- 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示了具有 Map 狀態範例專案之「動態」處理資料的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀

態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

  > **Tip**
  >
  > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

  準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

  建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

  建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

  > **Important**
  >
  > CloudFormation 範本中使用的每項服務可能會收取標準費用。

範例專案的資源部署完成後，您必須在執行狀態機器之前，將項目新增到 Amazon SQS 佇列並訂閱 Amazon SNS 主題。

步驟 2：訂閱 Amazon SNS 主題

1. 開啟 Amazon SNS 主控台。
2. 選擇 Topics (主題)，然後選擇 `Map` 狀態範例專案所建立的主題。

   該名稱將類似於-SNTOPIC-1CQO4HQ3 MapSampleProjIR1KN。
3. 選擇建立訂閱。

Create subscription (建立訂閱) 頁面隨即顯示，並列出主題的 Topic ARN (主題 ARN)。

4. 在 Protocol (通訊協定) 下方，選擇 Email (電子郵件)。
5. 在 Endpoint (端點) 下方，輸入電子郵件地址來訂閱主題。
6. 選擇建立訂閱。

> **Note**
>
> 您必須在電子郵件中確認訂閱，訂閱才會生效。

7. 開啟相關帳戶中的 Subscription Confirmation (訂閱確認) 電子郵件，接著開啟 Confirm subscription (確認訂閱) URL。

   即會顯示 Subscription confirmed! (訂閱確認！) 頁面。

步驟 3：將訊息新增至 Amazon SQS 佇列

1. 開啟 Amazon SQS 主控台。
2. 選擇 `Map` 狀態範例專案所建立的佇列。

   該名稱將類似於 MapSampleProj-SQUUE- 1vzdorn7。
3. 選擇傳送及接收訊息。
4. 在 [傳送和接收訊息] 頁面上，輸入訊息並選擇 [傳送訊息]。
5. 輸入其他訊息並選擇 [傳送訊息]。繼續輸入更多訊息，直到 Amazon SQS 佇列中有多則訊息為止。

步驟 4：運行狀態機

> **Note**
>
> Amazon SNS 中的佇列最終是一致的。為了獲得最佳結果，請在填入佇列和執行狀態機器之間等待幾分鐘。

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。

3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

      > **Note**
      >
      > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

   3. 選擇 Start execution (開始執行)。
   4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

狀態機代碼示例

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 Amazon SQS、Amazon SNS 和 Lambda 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN) 並傳遞`Parameters`至服務 API 來控制 Lambda、DynamoDB、Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
```

```
 "Comment": "An example of the Amazon States Language for reading messages from an SQS
queue and iteratively processing each message.",
 "StartAt": "Read messages from SQS Queue",
 "States": {
   "Read messages from SQS Queue": {
     "Type": "Task",
     "Resource": "arn:aws:states:::lambda:invoke",
     "OutputPath": "$.Payload",
     "Parameters": {
       "FunctionName": "MapSampleProj-ReadFromSQSQueueLambda-1MY3M63RMJVA9"
     },
     "Next": "Are there messages to process?"
   },
   "Are there messages to process?": {
     "Type": "Choice",
     "Choices": [
       {
         "Variable": "$",
         "StringEquals": "No messages",
         "Next": "Finish"
       }
     ],
     "Default": "Process messages"
   },
   "Process messages": {
     "Type": "Map",
     "Next": "Finish",
     "ItemsPath": "$",
     "Parameters": {
       "MessageNumber.$": "$$.Map.Item.Index",
       "MessageDetails.$": "$$.Map.Item.Value"
     },
     "Iterator": {
       "StartAt": "Write message to DynamoDB",
       "States": {
         "Write message to DynamoDB": {
           "Type": "Task",
           "Resource": "arn:aws:states:::dynamodb:putItem",
           "ResultPath": null,
           "Parameters": {
             "TableName": "MapSampleProj-DDBTable-YJDJ1MKIN6C5",
             "ReturnConsumedCapacity": "TOTAL",
             "Item": {
               "MessageId": {
```

```
                  "S.$": "$.MessageDetails.MessageId"
                },
                "Body": {
                  "S.$": "$.MessageDetails.Body"
                }
              }
            },
            "Next": "Remove message from SQS queue"
          },
          "Remove message from SQS queue": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "InputPath": "$.MessageDetails",
            "ResultPath": null,
            "Parameters": {
              "FunctionName": "MapSampleProj-DeleteFromSQSQueueLambda-198J2839ZO5K2",
              "Payload": {
                "ReceiptHandle.$": "$.ReceiptHandle"
              }
            },
            "Next": "Publish message to SNS topic"
          },
          "Publish message to SNS topic": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sns:publish",
            "InputPath": "$.MessageDetails",
            "Parameters": {
              "Subject": "Message from Step Functions!",
              "Message.$": "$.Body",
              "TopicArn": "arn:aws:sns:us-east-1:012345678910:MapSampleProj-
SNSTopic-1CQO4HQ3IR1KN"
            },
            "End": true
          }
        }
      }
    },
    "Finish": {
      "Type": "Succeed"
    }
  }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:012345678901:function:MapSampleProj-
ReadFromSQSQueueLambda-1MY3M63RMJVA9",
                "arn:aws:lambda:us-east-1:012345678901:function:MapSampleProj-
DeleteFromSQSQueueLambda-198J2839ZO5K2"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "dynamodb:PutItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:012345678901:table/MapSampleProj-DDBTable-
YJDJ1MKIN6C5"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:012345678901:MapSampleProj-
SNSTopic-1CQO4HQ3IR1KN"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

使用分散式地圖處理 CSV 檔案

此範例專案示範如何使用「分散式地圖」狀態來重複使用Lambda函數產生的 CSV 檔案的 10,000 列。CSV 檔案包含客戶訂單的運送資訊，並存放在 Amazon S3 儲存貯體中。分散式地圖會在 CSV 檔案中重複執行 10 列的批次，以進行資料分析。

分佈式地圖包含檢測任何延遲訂單的Lambda功能。分佈式映射還包含一個內聯映射，用於處理批處理延遲訂單，並在數組中返回這些延遲訂單。對於每個延遲的訂單，內聯映射將消息發送到Amazon SQS隊列。最後，此範例專案將地圖執行結果存 Amazon S3 到您的 AWS 帳戶.

使用分散式地圖，您一次最多可以執行 10,000 個 parallel 子工作流程執行。在此範例專案中，分散式地圖的最大並行設定為 1000，將其限制為 1000 個 parallel 子工作流程執行。

此範例專案會建立狀態機器、支援 AWS 資源，並設定相關的 IAM 許可。探索此範例專案，瞭解如何使用分散式地圖來協調大規模的 parallel 工作負載，或將其用作您自己專案的起點。

AWS CloudFormation 模板和其他資源

您可以使用 CloudFormation 範本來部署此範例專案。此範本會在您的中建立下列資源 AWS 帳戶：

- Step Functions 狀態機。
- 狀態機器的執行角色。此角色授予狀態機器存取其他資源所需的權限 AWS 服務 ，例如 Lambda 函數的叫用動作。
- 名為的 Lambda 函數`CSVGeneratorFunction`，可產生包含客戶訂單詳細資料的 CSV 檔案。
- CSV 產生器 Lambda 函數的執行角色。此角色授與存取其他的函數權限 AWS 服務。
- 用於存放產生的 CSV 檔案的 Amazon S3 輸入儲存貯體。
- 延遲訂單偵測 Lambda 函數，可分析 CSV 檔案資料並偵測任何延遲訂單。
- 延遲訂單 Lambda 函數的執行角色。此角色授與存取其他的函數權限 AWS 服務。
- Amazon S3 輸出儲存貯體，用於存放客戶訂單的分析結果。
- 一個 Amazon SQS 佇列，Step Functions 會針對每個延遲的訂單傳送訊息。這些訊息包含客戶及其訂單的 ID。
- 儲存與狀態機器執行歷程 CloudWatch 記錄相關資訊的記錄群組。

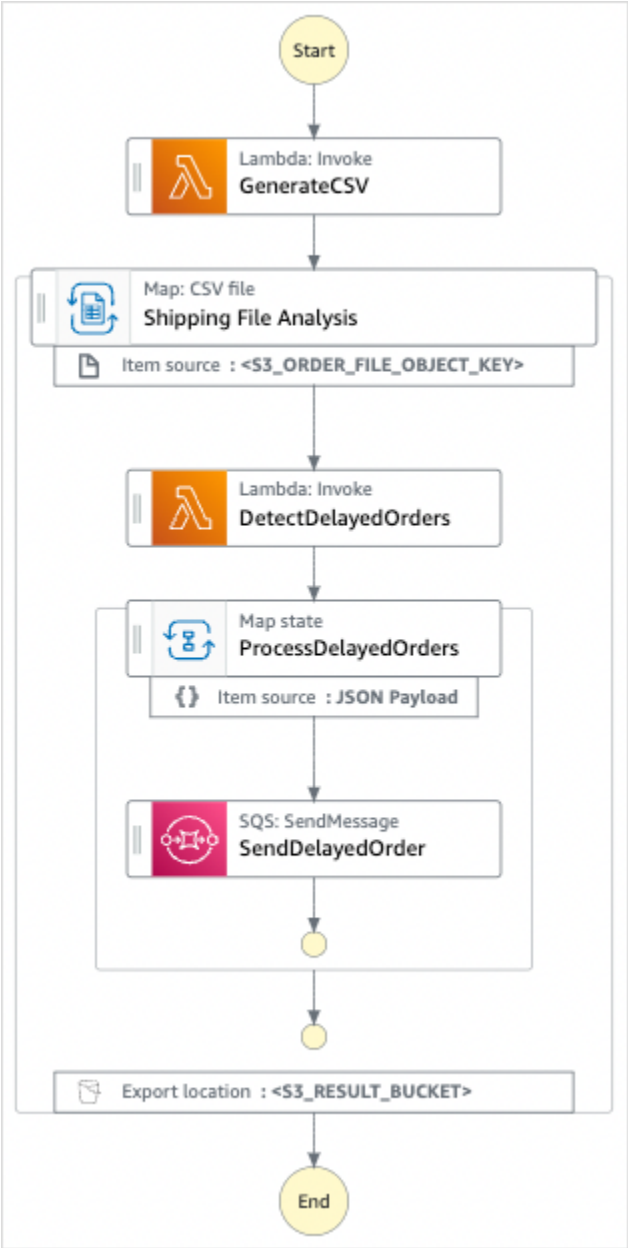
> **Important**
>
> 每項服務均需收取標準費用。

步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Distributed Map to process a CSV file in S3`**在搜尋方塊中輸入，然後選擇「分散式對應」，從傳回的搜尋結果處理 S3 中的 CSV 檔案。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   如需將為此範例專案建立之資源的相關資訊，請參閱AWS CloudFormation 模板和其他資源。

   下圖顯示了分散式地圖在 S3 範例專案中處理 CSV 檔案的工作流程圖：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

  > **Tip**
  >
  > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

  準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

  建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

  建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

  > **Important**
  >
  > CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

佈建和部署所有資源之後，您可以執行狀態機器。

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   a. (選擇性) 以 JSON 格式輸入輸入值以執行範例專案。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

b. 選擇 Start execution (開始執行)。

c. (選擇性)「Step Functions」主控台會將您導向標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

執行完成後，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。

- 如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。
- 如需有關在主控台中檢視分散式地圖狀態執行的詳細資訊，請參閱檢查地圖運行。

d. (選擇性) 檢閱匯出至 Amazon S3 儲存貯體的執行結果。這些結果包括資料，例如執行輸入和輸出、ARN 和執行狀態。如需更多詳細資訊，請參閱 ResultWriter。

使用分散式地圖處理 Amazon S3 儲存貯體中的資料

此範例專案示範如何使用「分散式地圖」狀態來處理大型資料，例如分析歷史氣象資料，以及識別每月地球平均溫度最高的氣象站。氣象資料會記錄在超過 12,000 個 CSV 檔案中，而這些檔案又會儲存在 Amazon S3 儲存貯體中。

此範例專案包含兩個名為分散式 S3 副本 NOA 資料和處理程序 NoAadata 的分散式地圖狀態。分散式 S3 副本 NOA 資料會重複執行名為的公有 Amazon S3 儲存貯體中的 CSV 檔案，noaa-gsod-pds並將它們複製到您的. AWS 帳戶 處理 Aadata 會重複複製的檔案，並包含執行溫度分析的 Lambda 函數。

範例專案會先透過呼叫 ListObjectsV2 API 動作來檢查 Amazon S3 儲存貯體的內容。根據回應此呼叫所傳回的金鑰數目，範例專案會執行下列其中一項決定：

- 如果金鑰計數大於或等於 1，專案會轉換為處理程序 A adata 狀態。此「分散式地圖」狀態包含一TemperatureFunction個名為的Lambda函數，可尋找每月平均溫度最高的氣象站。此函數會傳回`year-month`一個字典，其中包含氣象站做為值的資訊的字典。
- 如果傳回的金鑰計數不超過 1，[分散式 S3 複製 NOA Data] 狀態會列出公用儲存貯體中的所有物件，noaa-gsod-pds並以 100 個批次反覆方式將個別物件複製到帳戶中的另一個儲存貯體。內聯映射執行對象的迭代複製。

複製所有物件之後，專案會轉換為 ProcessNoAadata 狀態，以便處理氣象資料。

示例項目最終轉換為減速器Lambda函數，該函數對TemperatureFunction函數返回的結果執行最終聚合，並將結果寫入Amazon DynamoDB表中。

使用分散式地圖，您一次最多可以執行 10,000 個 parallel 子工作流程執行。在此範例專案中，處理程序 Aadata 分散式地圖的最大並行性設定為 3000，將其限制為 3000 個 parallel 子工作流程執行。

此範例專案會建立狀態機器、支援 AWS 資源，並設定相關的 IAM 許可。探索此範例專案，瞭解如何使用分散式地圖來協調大規模的 parallel 工作負載，或將其用作您自己專案的起點。

> **Important**
>
> 此範例專案僅在美國東部 (維吉尼亞北部) 區域提供。

AWS CloudFormation 模板和其他資源

您可以使用 CloudFormation 範本來部署此範例專案。此範本會在您的中建立下列資源 AWS 帳戶：

- Step Functions 狀態機。
- 狀態機器的執行角色。此角色授予狀態機器存取其他資源所需的權限 AWS 服務 ，例如 Lambda 函數的叫用動作。
- 一個名為的 Amazon S3 桶`N0AADataBucket`。此值區包含含有氣象資料的 CSV 檔案。
- 命名為的 Lambda 函數`ReducerFunction`，可執行氣象資料的最終彙總，並將結果寫入 Amazon DynamoDB 表格。
- 減速器 Lambda 函數的執行角色。此角色授與存取其他的函數權限 AWS 服務。
- 名為`ResultsBucket`用於存放天氣分析結果的 Amazon S3 輸出儲存貯體。
- 名為的 DynamoDB 表格`ResultsDynamoDBTable`，其中包含由. `ReducerFunction`
- 命名為的 Lambda 函數`TemperatureFunction`，用於查找最高的每月平均溫度。
- Lambda 函數的執行角色。此角色授與存取其他的函數權限 AWS 服務。
- 儲存與狀態機器執行歷程 CloudWatch 記錄相關資訊的記錄群組。

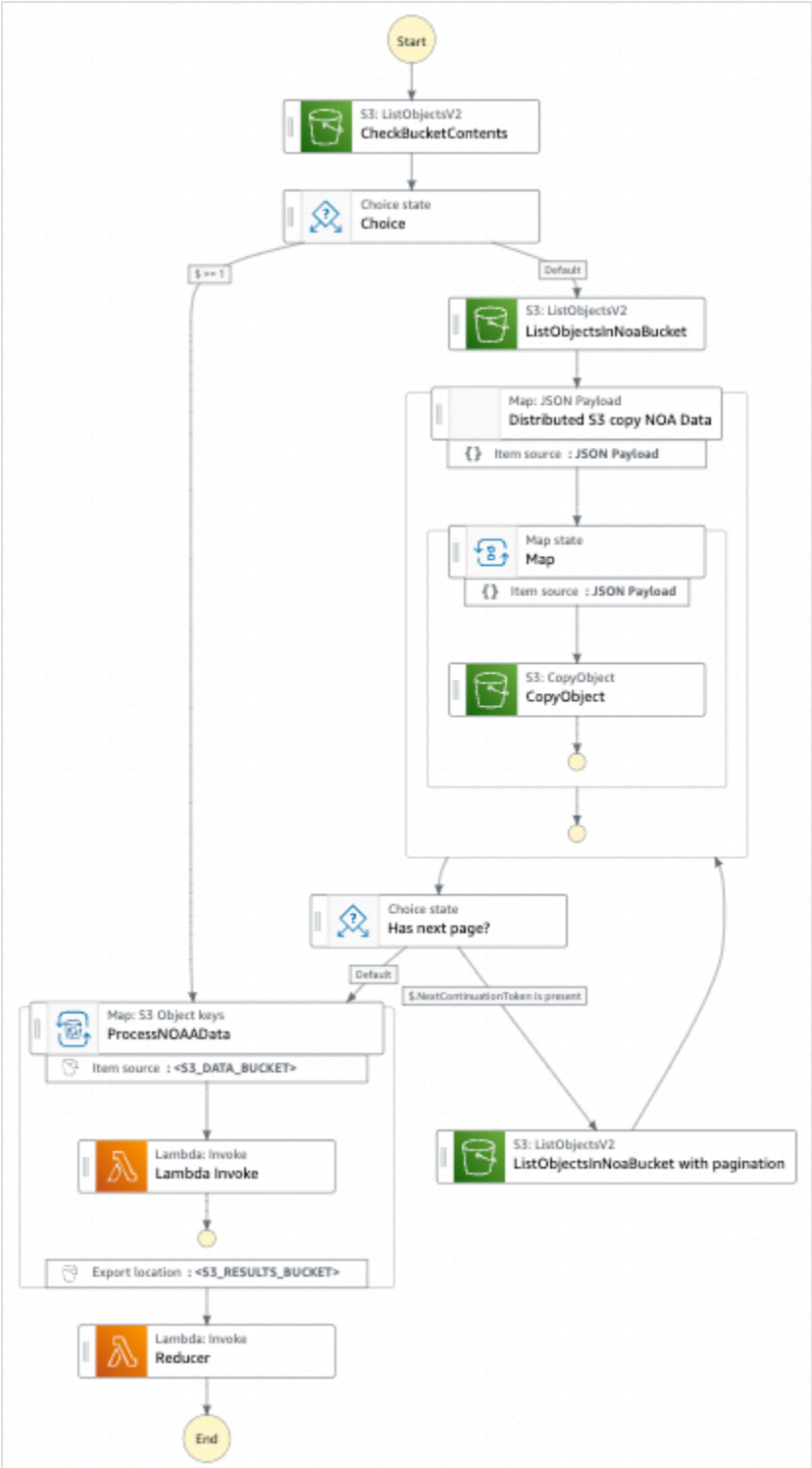
> **Important**
>
> 每項服務均需收取標準費用。

步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Distributed Map to process files in S3`**在搜尋方塊中輸入，然後選擇「分散式對應」，從傳回的搜尋結果中處理 S3 中的檔案。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   如需將為此範例專案建立之資源的相關資訊，請參閱AWS CloudFormation 模板和其他資源。

   下圖顯示了分散式地圖在 S3 範例專案中處理檔案的工作流程圖：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：
   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

佈建和部署所有資源後，您可以執行狀態機器。

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

a. (選擇性) 以 JSON 格式輸入輸入值以執行範例專案。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

b. 選擇 Start execution (開始執行)。

c. (選擇性)「Step Functions」主控台會將您導向標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   執行完成後，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。

   - 如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。
   - 如需有關在主控台中檢視分散式地圖狀態執行的詳細資訊，請參閱檢查地圖運行。

d. (選擇性) 檢閱匯出至 Amazon S3 儲存貯體的執行結果。這些結果包括資料，例如執行輸入和輸出、ARN 和執行狀態。如需更多詳細資訊，請參閱 ResultWriter。

訓練機器學習模型

此範例專案示範如何使用 SageMaker 和 AWS Step Functions 訓練機器學習模型，以及如何批次轉換測試資料集。

在此專案中，Step Functions 數使用 Lambda 函數來植入含有測試資料集的 Amazon S3 儲存貯體。然後，它會訓練機器學習模型，並使用SageMaker 服務整合執行批次轉換。

如需有關 SageMaker 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- SageMaker 使用 Step Functions 管理

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱SageMaker 定價。

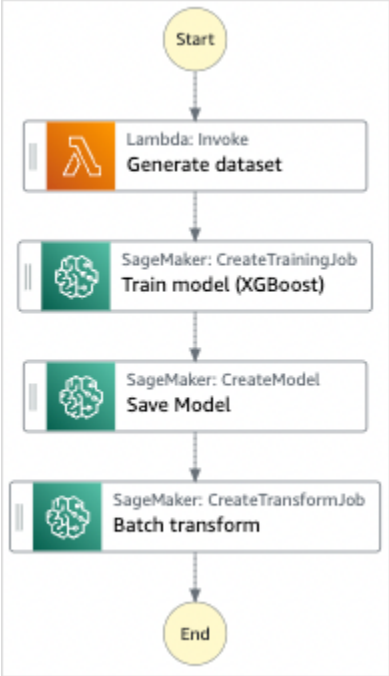
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Train a machine learning model`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [訓練機器學習模型]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇「執行示範」或「在其上建置」。

   此範例專案會部署下列資源：

   - 一個 AWS Lambda 函數
   - 一個 Amazon Simple Storage Service (Amazon S3) 儲存貯體
   - 一个 AWS Step Functions 状态机
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示「訓練機器學習模型」範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

     > **Tip**
     >
     > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

      > **Note**
      >
      > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

   3. 選擇 Start execution (開始執行)。

4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器 SageMaker 與 AWS Lambda 這些資源直接整合並傳遞參數，並將 Amazon S3 儲存貯體用於訓練資料來源和輸出。

瀏覽此範例狀態機器，以瞭解 Step Functions 如何控制 Lambda 和 SageMaker.

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "StartAt": "Generate dataset",
  "States": {
    "Generate dataset": {
      "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:TrainAndBatchTransform-SeedingFunction-17RNSOTG97HPV",
      "Type": "Task",
      "Next": "Train model (XGBoost)"
    },
    "Train model (XGBoost)": {
      "Resource": "arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters": {
        "AlgorithmSpecification": {
          "TrainingImage": "433757028032.dkr.ecr.us-west-2.amazonaws.com/
xgboost:latest",
          "TrainingInputMode": "File"
        },
        "OutputDataConfig": {
          "S3OutputPath": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/models"
        },
        "StoppingCondition": {
          "MaxRuntimeInSeconds": 86400
        },
        "ResourceConfig": {
          "InstanceCount": 1,
```

```
          "InstanceType": "ml.m4.xlarge",
          "VolumeSizeInGB": 30
        },
        "RoleArn": "arn:aws:iam::123456789012:role/TrainAndBatchTransform-
SageMakerAPIExecutionRole-Y9IX3DLF6EUO",
        "InputDataConfig": [
          {
            "DataSource": {
              "S3DataSource": {
                "S3DataDistributionType": "ShardedByS3Key",
                "S3DataType": "S3Prefix",
                "S3Uri": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/csv/
train.csv"
              }
            },
            "ChannelName": "train",
            "ContentType": "text/csv"
          }
        ],
        "HyperParameters": {
          "objective": "reg:logistic",
          "eval_metric": "rmse",
          "num_round": "5"
        },
        "TrainingJobName.$": "$$.Execution.Name"
      },
      "Type": "Task",
      "Next": "Save Model"
    },
    "Save Model": {
      "Parameters": {
        "PrimaryContainer": {
          "Image": "433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest",
          "Environment": {},
          "ModelDataUrl.$": "$.ModelArtifacts.S3ModelArtifacts"
        },
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/TrainAndBatchTransform-
SageMakerAPIExecutionRole-Y9IX3DLF6EUO",
        "ModelName.$": "$.TrainingJobName"
      },
      "Resource": "arn:aws:states:::sagemaker:createModel",
      "Type": "Task",
      "Next": "Batch transform"
    },
```

```
    "Batch transform": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createTransformJob.sync",
      "Parameters": {
        "ModelName.$": "$$.Execution.Name",
        "TransformInput": {
          "CompressionType": "None",
          "ContentType": "text/csv",
          "DataSource": {
            "S3DataSource": {
              "S3DataType": "S3Prefix",
              "S3Uri": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/csv/
test.csv"
            }
          }
        },
        "TransformOutput": {
          "S3OutputPath": "s3://trainandbatchtransform-s3bucket-1jn1le6gadwfz/output"
        },
        "TransformResources": {
          "InstanceCount": 1,
          "InstanceType": "ml.m4.xlarge"
        },
        "TransformJobName.$": "$$.Execution.Name"
      },
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

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

```
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

下列政策允許 Lambda 函數使用範例資料植入 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

調校機器學習模型

此範例專案示範 SageMaker 如何調整機器學習模型的超參數，以及批次轉換測試資料集。

在此專案中，Step Functions 數使用 Lambda 函數來植入含有測試資料集的 Amazon S3 儲存貯體。然後，它會使用SageMaker服務整合建立超參數調整工作。然後，它會使用 Lambda 函數擷取資料路徑、儲存調整模型、擷取模型名稱，然後執行批次轉換工作以在中 SageMaker執行推論。

如需有關 SageMaker 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- SageMaker 使用 Step Functions 管理

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱SageMaker定價。

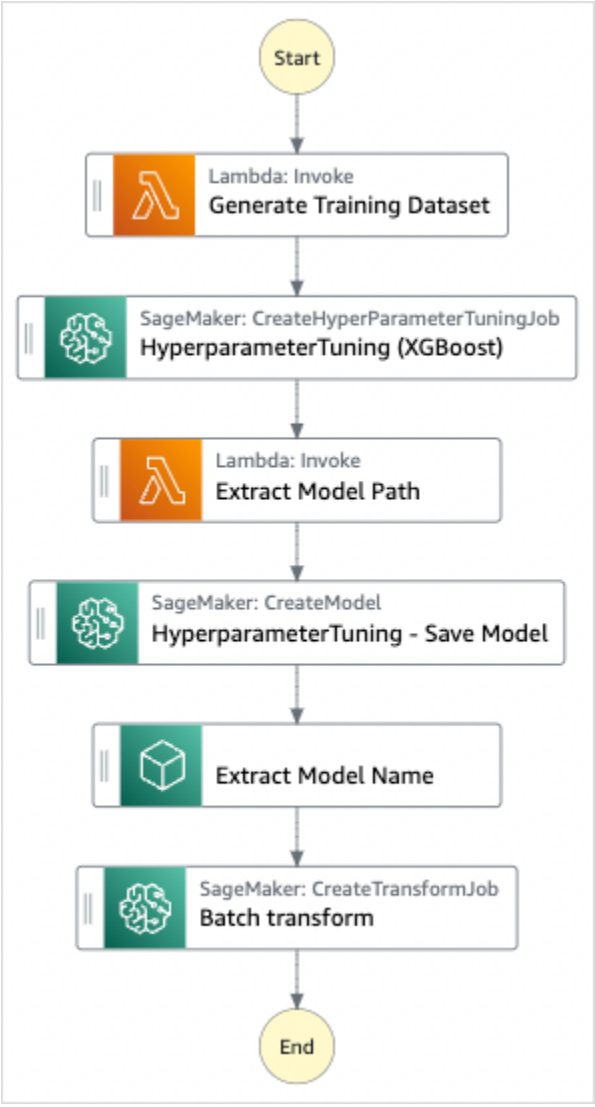
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Tune a machine learning model`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇「調整機器學習模型」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 三大 AWS Lambda 功能
   - 一個 Amazon Simple Storage Service (Amazon S3) 儲存貯體
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下圖顯示了「調整機器學習模型」範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. 「Step Functions」主控台會將您引導至標題為您的執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器 SageMaker 與 AWS Lambda 這些資源直接整合並傳遞參數，並將 Amazon S3 儲存貯體用於訓練資料來源和輸出。

瀏覽此範例狀態機器，以瞭解 Step Functions 如何控制 Lambda 和 SageMaker.

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "StartAt": "Generate Training Dataset",
    "States": {
        "Generate Training Dataset": {
            "Resource": "arn:aws:lambda:us-
west-2:012345678912:function:StepFunctionsSample-SageMa-
LambdaForDataGeneration-1TF67BUE5A12U",
```

```
            "Type": "Task",
            "Next": "HyperparameterTuning (XGBoost)"
        },
        "HyperparameterTuning (XGBoost)": {
            "Resource":
 "arn:aws:states:::sagemaker:createHyperParameterTuningJob.sync",
            "Parameters": {
                "HyperParameterTuningJobName.$": "$.body.jobName",
                "HyperParameterTuningJobConfig": {
                    "Strategy": "Bayesian",
                    "HyperParameterTuningJobObjective": {
                        "Type": "Minimize",
                        "MetricName": "validation:rmse"
                    },
                    "ResourceLimits": {
                        "MaxNumberOfTrainingJobs": 2,
                        "MaxParallelTrainingJobs": 2
                    },
                    "ParameterRanges": {
                        "ContinuousParameterRanges": [{
                                "Name": "alpha",
                                "MinValue": "0",
                                "MaxValue": "1000",
                                "ScalingType": "Auto"
                            },
                            {
                                "Name": "gamma",
                                "MinValue": "0",
                                "MaxValue": "5",
                                "ScalingType": "Auto"
                            }
                        ],
                        "IntegerParameterRanges": [{
                                "Name": "max_delta_step",
                                "MinValue": "0",
                                "MaxValue": "10",
                                "ScalingType": "Auto"
                            },
                            {
                                "Name": "max_depth",
                                "MinValue": "0",
                                "MaxValue": "10",
                                "ScalingType": "Auto"
                            }
```

```
                        ]
                    }
                },
                "TrainingJobDefinition": {
                    "AlgorithmSpecification": {
                        "TrainingImage": "433757028032.dkr.ecr.us-west-2.amazonaws.com/
xgboost:latest",
                        "TrainingInputMode": "File"
                    },
                    "OutputDataConfig": {
                        "S3OutputPath": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/models"
                    },
                    "StoppingCondition": {
                        "MaxRuntimeInSeconds": 86400
                    },
                    "ResourceConfig": {
                        "InstanceCount": 1,
                        "InstanceType": "ml.m4.xlarge",
                        "VolumeSizeInGB": 30
                    },
                    "RoleArn": "arn:aws:iam::012345678912:role/StepFunctionsSample-
SageM-SageMakerAPIExecutionRol-1MNH1VS5CGGOG",
                    "InputDataConfig": [{
                        "DataSource": {
                            "S3DataSource": {
                                "S3DataDistributionType": "FullyReplicated",
                                "S3DataType": "S3Prefix",
                                "S3Uri": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/csv/train.csv"
                            }
                        },
                        "ChannelName": "train",
                        "ContentType": "text/csv"
                    },
                    {
                        "DataSource": {
                            "S3DataSource": {
                                "S3DataDistributionType": "FullyReplicated",
                                "S3DataType": "S3Prefix",
                                "S3Uri": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/csv/validation.csv"
                            }
                        },
```

```
                        "ChannelName": "validation",
                        "ContentType": "text/csv"
                    }],
                    "StaticHyperParameters": {
                        "precision_dtype": "float32",
                        "num_round": "2"
                    }
                }
            },
            "Type": "Task",
            "Next": "Extract Model Path"
        },
        "Extract Model Path": {
            "Resource": "arn:aws:lambda:us-
west-2:012345678912:function:StepFunctionsSample-SageM-LambdaToExtractModelPath-
V0R37CVARUS9",
            "Type": "Task",
            "Next": "HyperparameterTuning - Save Model"
        },
        "HyperparameterTuning - Save Model": {
            "Parameters": {
                "PrimaryContainer": {
                    "Image": "433757028032.dkr.ecr.us-west-2.amazonaws.com/
xgboost:latest",
                    "Environment": {},
                    "ModelDataUrl.$": "$.body.modelDataUrl"
                },
                "ExecutionRoleArn": "arn:aws:iam::012345678912:role/
StepFunctionsSample-SageM-SageMakerAPIExecutionRol-1MNH1VS5CGGOG",
                "ModelName.$": "$.body.bestTrainingJobName"
            },
            "Resource": "arn:aws:states:::sagemaker:createModel",
            "Type": "Task",
            "Next": "Extract Model Name"
        },
        "Extract Model Name": {
            "Resource": "arn:aws:lambda:us-
west-2:012345678912:function:StepFunctionsSample-SageM-
LambdaToExtractModelName-8FUOB30SM5EM",
            "Type": "Task",
            "Next": "Batch transform"
        },
        "Batch transform": {
            "Type": "Task",
```

```
            "Resource": "arn:aws:states:::sagemaker:createTransformJob.sync",
            "Parameters": {
                "ModelName.$": "$.body.jobName",
                "TransformInput": {
                    "CompressionType": "None",
                    "ContentType": "text/csv",
                    "DataSource": {
                        "S3DataSource": {
                            "S3DataType": "S3Prefix",
                            "S3Uri": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/csv/test.csv"
                        }
                    }
                },
                "TransformOutput": {
                    "S3OutputPath": "s3://stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/output"
                },
                "TransformResources": {
                    "InstanceCount": 1,
                    "InstanceType": "ml.m4.xlarge"
                },
                "TransformJobName.$": "$.body.jobName"
            },
            "End": true
        }
    }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

下列 IAM 政策已附加至狀態機器，並允許狀態機器執行存取必要 SageMaker的 Lambda 和 Amazon S3 資源。

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

```
        {
            "Action": [
                "sagemaker:CreateHyperParameterTuningJob",
                "sagemaker:DescribeHyperParameterTuningJob",
                "sagemaker:StopHyperParameterTuningJob",
                "sagemaker:ListTags",
                "sagemaker:CreateModel",
                "sagemaker:CreateTransformJob",
                "iam:PassRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageMa-LambdaForDataGeneration-1TF67BUE5A12U",
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageM-LambdaToExtractModelPath-V0R37CVARUS9",
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageM-LambdaToExtractModelName-8FUOB30SM5EM"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule",
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule",
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTuningJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
```

```
}
```

下列 IAM 政策會在HyperparameterTuning州/省的TrainingJobDefinition和HyperparameterTuning欄位中參照。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "sagemaker:DescribeHyperParameterTuningJob",
                "sagemaker:StopHyperParameterTuningJob",
                "sagemaker:ListTags"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f",
            "Effect": "Allow"
        }
```

```
    ]
}
```

下列 IAM 政策允許 Lambda 函數使用範例資料植入 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

處理來自 Amazon SQS 的大量訊息 (快速工作流程)

此範例專案示範如何使用 AWS Step Functions 快速工作流程處理來自大量事件來源的訊息或資料，例如 Amazon Simple Queue Service (Amazon SQS)。由於快速工作流程能以非常高的速率啟動，因此非常適用於大量事件處理或串流資料工作負載。

以下是從事件來源執行狀態機器的兩種常用方法：

- 設定 Amazon CloudWatch 事件規則，以便在事件來源發出事件時啟動狀態機器執行。如需詳細資訊，請參閱建立在 CloudWatch 事件上觸發的事件規則。
- 將事件來源映射至 Lambda 函數，然後撰寫函數程式碼以執行您的狀態機器。每次事件來源發出事件時，都會叫用 AWS Lambda 函數，進而啟動狀態機器執行。如需詳細資訊，請參閱AWS Lambda 搭配 Amazon SQS 使用。

此範例專案會在每次 Amazon SQS 佇列傳送訊息時，使用第二種方法啟動執行。您可以使用類似的組態，從其他事件來源觸發快速工作流程執行，例如 Amazon 簡單儲存服務 (Amazon S3)、Amazon DynamoDB 和 Amazon Kinesis。

如需有關 Express 工作流程和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- 標準與快速工作流程
- AWS Step Functions 搭配其他服務使用
- 配額

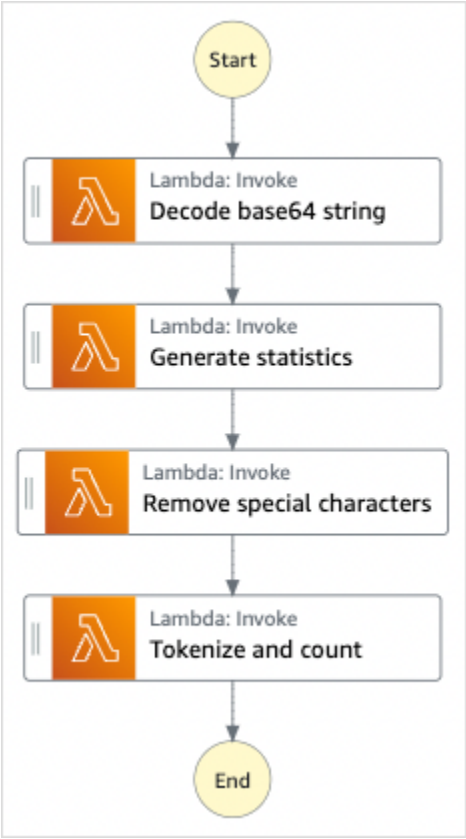
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Process high-volume messages from SQS`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇「處理來自 SQS 的大量訊息」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 四 Lambda 数
   - Amazon SQS 隊列
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下列影像顯示處理來自 SQS 範例專案的大量訊息的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：觸發狀態機執行

1. 開啟 Amazon SQS 主控台。
2. 選取由範例專案建立的佇列。

   該名稱將類似於 Example-SQSQueue-wJalrXUtnFEMI。
3. 在 Queue Actions (佇列動作) 清單中，選取 Send a Message (傳送訊息)。
4. 使用複製按鈕來複製以下訊息，然後在 Send a Message (傳送訊息) 視窗中，輸入該訊息，然後選擇 Send Message (傳送訊息)。

> **Note**
>
> 在此範例訊息中，`input:` 行已使用換行符號進行格式化以符合頁面。使用複製按鈕，或確保將該行輸入為不換行的單一行。

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc
```

```
  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw
  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT
  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU
  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic
  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB
  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBiZWF0aW5nIG1pb
}
```

5. 選擇關閉。
6. 開啟「Step Functions」主控台。
7. 轉到您的 Amazon CloudWatch 日誌日誌組並檢查日誌。記錄群組的名稱看起來像範例 ExpressLogGroup-如何使用。

範例 Lambda 函數代碼

以下是 Lambda 函數程式碼，顯示起始 Lambda 函數如何使用 AWS SDK 啟動狀態機器執行。

```
import boto3

def lambda_handler(event, context):
    message_body = event['Records'][0]['body']
    client = boto3.client('stepfunctions')
    response = client.start_execution(
        stateMachineArn='${ExpressStateMachineArn}',
        input=message_body
    )
```

範例狀態機器程式碼

此範例專案中的快速工作流程，包含了一組用於文字處理的 Lambda 函數。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
```

```
  "Comment": "An example of using Express workflows to run text processing for each
message sent from an SQS queue.",
  "StartAt": "Decode base64 string",
  "States": {
    "Decode base64 string": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<BASE64_DECODER_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Generate statistics"
    },
    "Generate statistics": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<TEXT_STATS_GENERATING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Remove special characters"
    },
    "Remove special characters": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<STRING_CLEANING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Tokenize and count"
    },
    "Tokenize and count": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<TOKENIZING_AND_WORD_COUNTING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "End": true
    }
```

```
  }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:example-
Base64DecodeLambda-wJalrXUtnFEMI",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
StringCleanerLambda-je7MtGbClwBF",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
TokenizerCounterLambda-wJalrXUtnFEMI",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
GenerateStatsLambda-je7MtGbClwBF"
            ],
            "Effect": "Allow"
        }
    ]
}
```

下列原則可確保記 CloudWatch 錄檔擁有足夠的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
```

```
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

選擇性檢查點範例 (快速工作流程)

此範例專案示範如何藉由執行會進行選擇性檢查點的模擬電子商務工作流程，來結合標準和快速工作流程。部署此範例專案會建立標準工作流程狀態機器、巢狀快速工作流程狀態機、 AWS Lambda 函數、Amazon Simple Queue Service (Amazon SQS) 佇列，以及 Amazon Simple Notification Service (Amazon SNS) 主題。

如需有關 Express 工作流程、巢狀工作流程和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- 標準與快速工作流程
- 從任務狀態開始工作流程執行
- AWS Step Functions 搭配其他服務使用

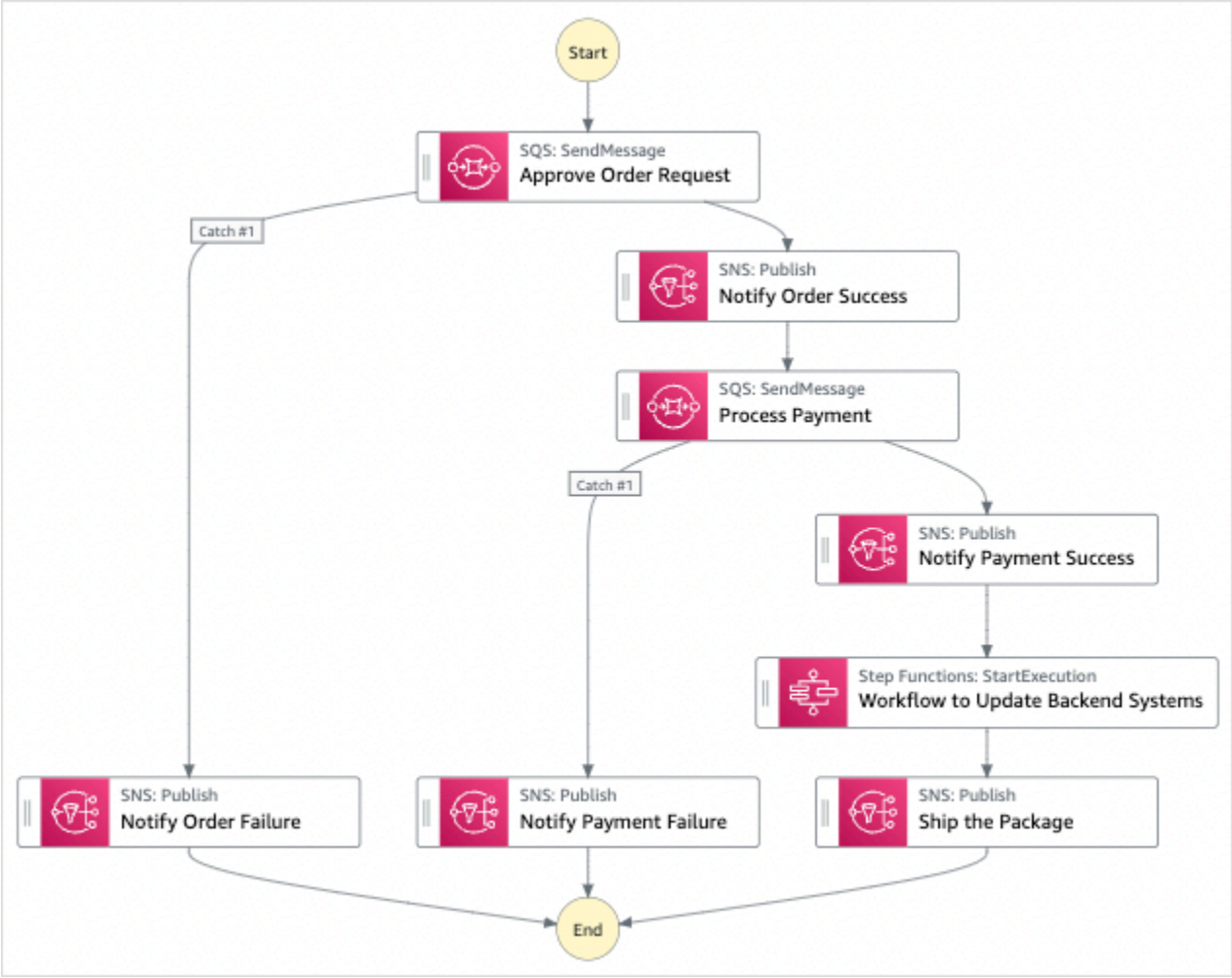
步驟 1：建立狀態機並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Selective checkpointing example`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇選擇性檢查點範例。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

此範例專案會部署下列資源：

- 一個 AWS Lambda 函數
- Amazon SQS 隊列
- Amazon SNS 主題
- 標準型的 AWS Step Functions 狀態機
- 快遞類型的 Step Functions 狀態機
- 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示了「選擇性檢查點範例」範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

- 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

  在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

  > **Important**
  >
  > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

  > **Tip**
  >
  > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

  準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

  建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

  建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

  > **Important**
  >
  > CloudFormation 範本中使用的每項服務可能會收取標準費用。

部署範例專案資源後，請執行以下作業。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。

2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

      > **Note**
      >
      > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

   3. 選擇 Start execution (開始執行)。
   4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

      若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。
4. 轉到您的CloudWatch 日誌日誌組並檢查日誌。記錄群組的名稱看起來像範例 ExpressLogGroup-如何使用。

父系的範例狀態機器程式碼 (標準工作流程)

此範例專案中的狀態機器與 Amazon SQS、Amazon SNS 和 Step Functions 快速工作流程整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何處理來自 Amazon SQS 和 Amazon SNS 的輸入，然後使用巢狀快速工作流程狀態機器更新後端系統。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "Comment": "An example of combining standard and express workflows to run a mock e-
commerce workflow that does selective checkpointing.",
    "StartAt": "Approve Order Request",
    "States": {
        "Approve Order Request": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sqs:sendMessage.waitForTaskToken",
            "Parameters": {
                "QueueUrl": "<SQS_QUEUE_URL>",
                "MessageBody": {
                    "MessageTitle": "Order Request received. Pausing workflow to wait
 for manual approval. ",
                    "TaskToken.$": "$$.Task.Token"
                }
            },
            "Next": "Notify Order Success",
            "Catch": [
                {
                    "ErrorEquals": [
                        "States.ALL"
                    ],
                    "Next": "Notify Order Failure"
                }
            ]
        },
        "Notify Order Success": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Order has been approved. Resuming workflow.",
                "TopicArn": "<SNS_ARN>"
            },
            "Next": "Process Payment"
        },
        "Notify Order Failure": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Order not approved. Order failed.",
                "TopicArn": "<SNS_ARN>"
            },
            "End": true
```

```
        },
        "Process Payment": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sqs:sendMessage.waitForTaskToken",
            "Parameters": {
                "QueueUrl": "<SQS_QUEUE_URL>",
                "MessageBody": {
                    "MessageTitle": "Payment sent to third-party for processing.
Pausing workflow to wait for response.",
                    "TaskToken.$": "$$.Task.Token"
                }
            },
            "Next": "Notify Payment Success",
            "Catch": [
                {
                    "ErrorEquals": [
                        "States.ALL"
                    ],
                    "Next": "Notify Payment Failure"
                }
            ]
        },
        "Notify Payment Success": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Payment processing succeeded. Resuming workflow.",
                "TopicArn": "<SNS_ARN>"
            },
            "Next": "Workflow to Update Backend Systems"
        },
        "Notify Payment Failure": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Payment processing failed.",
                "TopicArn": "<SNS_ARN>"
            },
            "End": true
        },
        "Workflow to Update Backend Systems": {
            "Comment": "Starting an execution of an Express workflow to handle backend
updates. Express workflows are fast and cost-effective for steps where checkpointing
isn't required.",
```

```
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::states:startExecution.sync",
            "Parameters": {
                "StateMachineArn": "<UPDATE_DATABASE_EXPRESS_STATE_MACHINE_ARN>",
                "Input": {
                    "AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
                }
            },
            "Next": "Ship the Package"
        },
        "Ship the Package": {
            "Type": "Task",
            "Resource": "arn:<PARTITION>:states:::sns:publish",
            "Parameters": {
                "Message": "Order and payment received, database is updated and the
 package is ready to ship.",
                "TopicArn": "<SNS_ARN>"
            },
            "End": true
        }
    }
}
```

父狀態機器的 IAM 角色範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

Amazon SNS 政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Amazon SQS 政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:123456789012:Checkpoint-SQSQueue-
je7MtGbClwBF",
            "Effect": "Allow"
        }
    ]
}
```

狀態執行政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "states:StartExecution",
                "states:DescribeExecution",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule",
            "Effect": "Allow"
        }
    ]
}
```

巢狀狀態機器的範例狀態機器程式碼 (快速工作流程)

此範例專案中的狀態機器，會在受到父狀態機器呼叫時更新後端資訊。

瀏覽此示例狀態機，以了解 Step Functions 如何更新模擬電子商務後端系統的不同組件。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。


```
{
  "StartAt": "Update Order History",
  "States": {
    "Update Order History": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update order history."
        }
      },
      "Next": "Update Data Warehouse"
    },
```

```
    "Update Data Warehouse": {
      "Type" : "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update data warehouse."
        }
      },
      "Next": "Update Customer Profile"
    },
    "Update Customer Profile": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update customer profile."
        }
      },
      "Next": "Update Inventory"
    },
    "Update Inventory": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Parameters": {
        "FunctionName": "Checkpoint-UpdateDatabaseLambdaFunction-wJalrXUtnFEMI",
        "Payload": {
          "Message": "Update inventory."
        }
      },
      "End": true
    }
  }
}
```

子狀態機器的 IAM 角色範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

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

```
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
            ],
            "Effect": "Allow"
        }
    ]
}
```

下列原則可確保 CloudWatch 記錄擁有足夠的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

建立 AWS CodeBuild 專案 (CodeBuildAmazon SNS)

此範例專案示範如何用 AWS Step Functions 來建置 AWS CodeBuild 專案、執行測試，然後傳送 Amazon SNS 通知。

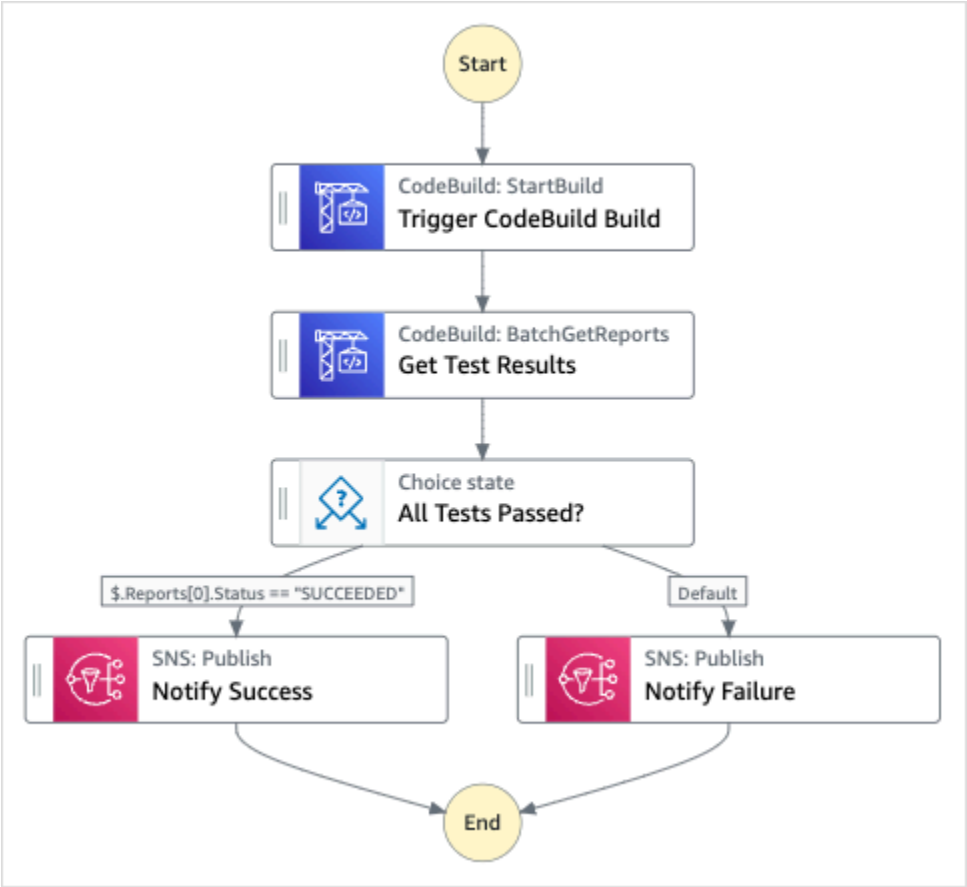
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Start a CodeBuild build`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [啟動 CodeBuild 組建]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 一個 AWS CodeBuild 構建
   - Amazon SNS 主題
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下列影像顯示 [啟動 CodeBuild 組建範例專案] 的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器與 CodeBuild 和 Amazon SNS 整合。

瀏覽此範例狀態機器，查看 Step Functions 如何使用狀態機器建立 CodeBuild 專案，然後傳送 Amazon SNS 主題，其中包含任務是成功還是失敗的訊息。

如需 Step Functions 如何控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of using CodeBuild to run tests, get test results and send a
 notification.",
  "StartAt": "Trigger CodeBuild Build",
  "States": {
    "Trigger CodeBuild Build": {
      "Type": "Task",
      "Resource": "arn:aws:states:::codebuild:startBuild.sync",
      "Parameters": {
        "ProjectName": "CodeBuildProject-Dtw1jBhEYGDf"
      },
      "Next": "Get Test Results"
    },
    "Get Test Results": {
      "Type": "Task",
      "Resource": "arn:aws:states:::codebuild:batchGetReports",
      "Parameters": {
        "ReportArns.$": "$.Build.ReportArns"
```

```
      },
      "Next": "All Tests Passed?"
    },
    "All Tests Passed?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.Reports[0].Status",
          "StringEquals": "SUCCEEDED",
          "Next": "Notify Success"
        }
      ],
      "Default": "Notify Failure"
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "CodeBuild build tests succeeded",
        "TopicArn": "arn:aws:sns:sa-east-1:123456789012:StepFunctionsSample-
CodeBuildExecution3da9ead6-bc1f-4441-99ac-591c140019c4-SNSTopic-EVYLVNGW85JP"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "CodeBuild build tests failed",
        "TopicArn": "arn:aws:sns:sa-east-1:123456789012:StepFunctionsSample-
CodeBuildExecution3da9ead6-bc1f-4441-99ac-591c140019c4-SNSTopic-EVYLVNGW85JP"
      },
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

預先處理資料並訓練機器學習模型

此範例專案示範如何使用 SageMaker 和預先處理資料， AWS Step Functions 以及如何訓練機器學習模型。

在此專案中，Step Functions 數使用 Lambda 函數來植入具有測試資料集和 Python 指令碼的 Amazon S3 儲存貯體以進行資料處理。然後，它會訓練機器學習模型，並使用SageMaker 服務整合執行批次轉換。

如需有關 SageMaker 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- SageMaker 使用 Step Functions 管理

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱SageMaker 定價。

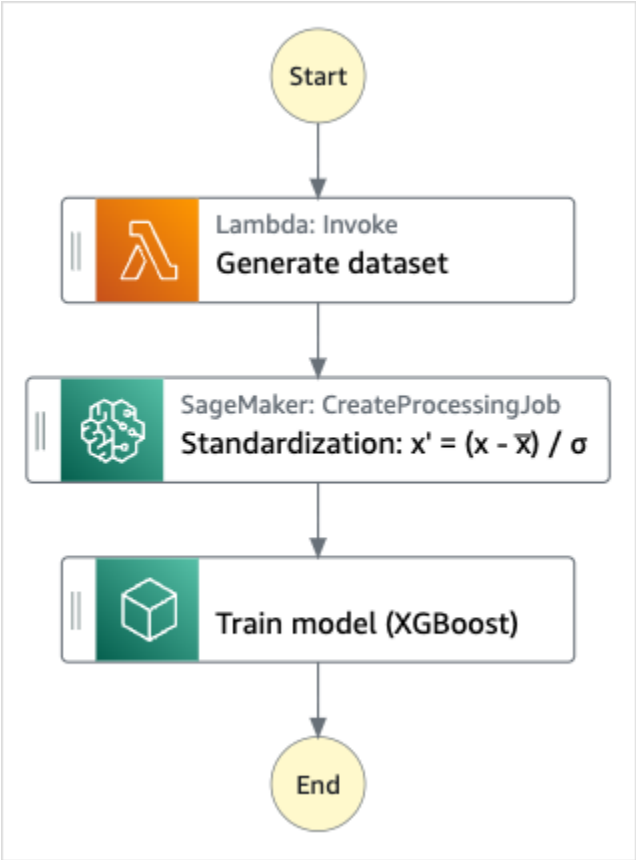
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Preprocess data and train a machine learning model`**在搜尋方塊中輸入，然後選擇「預處理資料」，並從傳回的搜尋結果中訓練機器學習模型。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 一個 AWS Lambda 函數
   - Amazon S3 儲存貯體
   - 一个 AWS Step Functions 状态机
   - 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示了「預處理」資料的工作流程圖形，並訓練機器學習模型範例專案：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器 SageMaker 與 AWS Lambda 這些資源直接整合並傳遞參數，並將 Amazon S3 儲存貯體用於訓練資料來源和輸出。

瀏覽此範例狀態機器，以瞭解 Step Functions 如何控制 Lambda 和 SageMaker.

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "StartAt": "Generate dataset",
  "States": {
    "Generate dataset": {
      "Resource": "arn:aws:lambda:sa-east-1:1234567890:function:FeatureTransform-
LambaForDataGeneration-17M8LX7IO9LUW",
      "Type": "Task",
      "Next": "Standardization: x' = (x - x̄) / σ"
    },
    "Standardization: x' = (x - x̄) / σ": {
      "Resource": "arn:aws:states:::sagemaker:createProcessingJob.sync",
      "Parameters": {
        "ProcessingResources": {
          "ClusterConfig": {
            "InstanceCount": 1,
            "InstanceType": "ml.m5.xlarge",
            "VolumeSizeInGB": 10
          }
```

```
        },
        "ProcessingInputs": [
            {
                "InputName": "input-1",
                "S3Input": {
                    "S3Uri": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz/
input/raw.csv",
                    "LocalPath": "/opt/ml/processing/input",
                    "S3DataType": "S3Prefix",
                    "S3InputMode": "File",
                    "S3DataDistributionType": "FullyReplicated",
                    "S3CompressionType": "None"
                }
            },
            {
                "InputName": "code",
                "S3Input": {
                    "S3Uri": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz/
code/transform.py",
                    "LocalPath": "/opt/ml/processing/input/code",
                    "S3DataType": "S3Prefix",
                    "S3InputMode": "File",
                    "S3DataDistributionType": "FullyReplicated",
                    "S3CompressionType": "None"
                }
            }
        ],
        "ProcessingOutputConfig": {
            "Outputs": [
                {
                    "OutputName": "train_data",
                    "S3Output": {
                        "S3Uri": "s3://featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/train",
                        "LocalPath": "/opt/ml/processing/output/train",
                        "S3UploadMode": "EndOfJob"
                    }
                }
            ]
        },
        "AppSpecification": {
            "ImageUri": "737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-
learn:0.20.0-cpu-py3",
            "ContainerEntrypoint": [
```

```
                "python3",
                "/opt/ml/processing/input/code/transform.py"
            ]
        },
        "StoppingCondition": {
            "MaxRuntimeInSeconds": 300
        },
        "RoleArn": "arn:aws:iam::1234567890:role/SageMakerAPIExecutionRole-
AIDACKCEVSQ6C2EXAMPLE",
        "ProcessingJobName.$": "$$.Execution.Name"
      },
      "Type": "Task",
      "Next": "Train model (XGBoost)"
    },
    "Train model (XGBoost)": {
      "Resource": "arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters": {
        "AlgorithmSpecification": {
          "TrainingImage": "855470959533.dkr.ecr.sa-east-1.amazonaws.com/
xgboost:latest",
          "TrainingInputMode": "File"
        },
        "OutputDataConfig": {
          "S3OutputPath": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz/
models"
        },
        "StoppingCondition": {
          "MaxRuntimeInSeconds": 86400
        },
        "ResourceConfig": {
          "InstanceCount": 1,
          "InstanceType": "ml.m5.xlarge",
          "VolumeSizeInGB": 30
        },
        "RoleArn": "arn:aws:iam::1234567890:role/SageMakerAPIExecutionRole-
AIDACKCEVSQ6C2EXAMPLE",
        "InputDataConfig": [
          {
            "DataSource": {
              "S3DataSource": {
                "S3DataDistributionType": "ShardedByS3Key",
                "S3DataType": "S3Prefix",
                "S3Uri": "s3://featuretransform-bucketforcodeanddata-1jn1le6gadwfz"
              }
```

```
            },
            "ChannelName": "train",
            "ContentType": "text/csv"
          }
        ],
        "HyperParameters": {
          "objective": "reg:logistic",
          "eval_metric": "rmse",
          "num_round": "5"
        },
        "TrainingJobName.$": "$$.Execution.Name"
      },
      "Type": "Task",
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
```

```
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

下列政策允許 Lambda 函數使用範例資料植入 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

Lambda 協調流程範例

此範例專案示範如何整合 Step AWS Lambda Functions 狀態機中的函式。

在此項目中，Step Functions 使用 Lambda 函數來檢查股票價格並確定買入或賣出交易建議。然後向用戶提供此建議，並可以選擇購買還是出售股票。使用 SNS 主題返回交易結果。

如需有關 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 以下項目的 IAM 政策：
  - 的 IAM 政策 AWS Lambda
  - Amazon SQS 的 IAM 政策

- Amazon SNS 的 IAM 政策

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱定價。

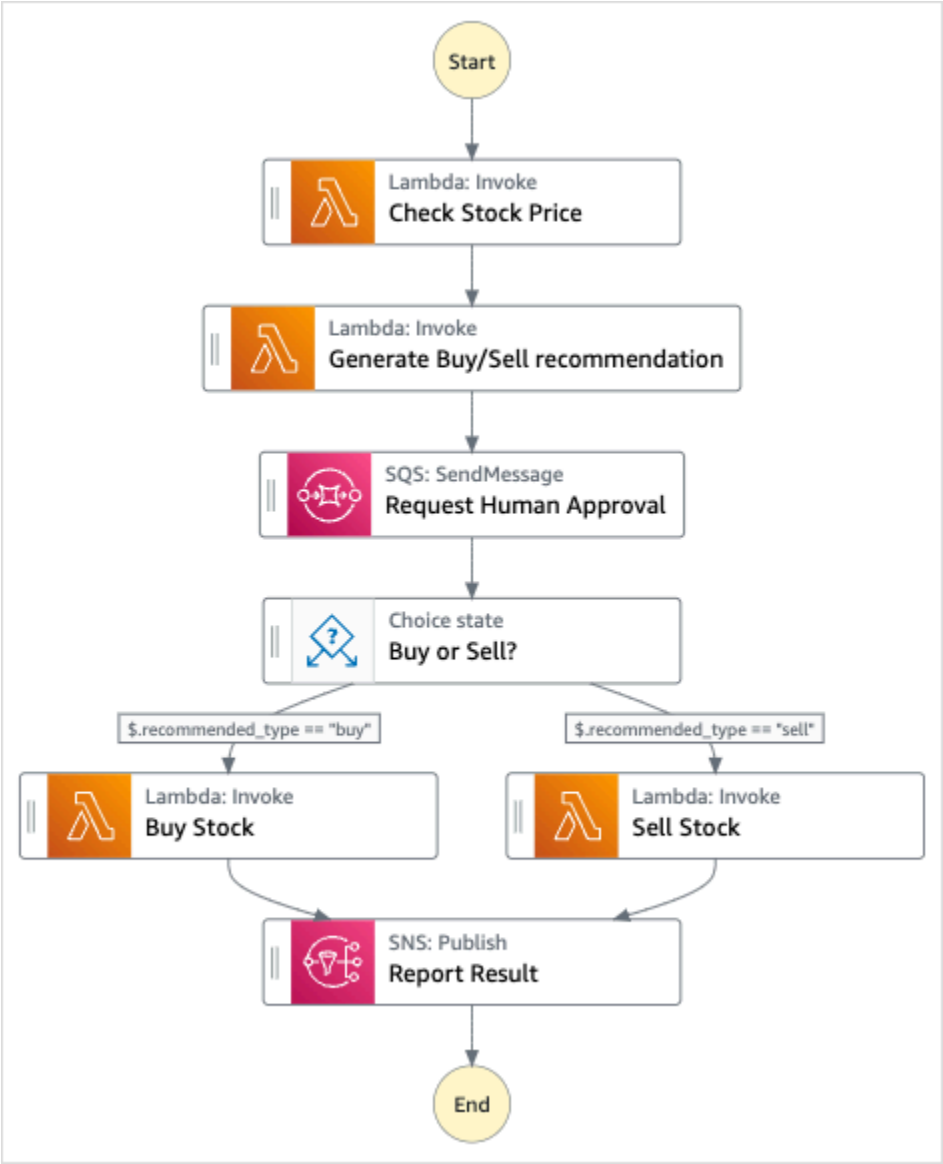
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Orchestrate Lambda functions`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 Orchestrate Lambda 函數。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 五大Lambda功能
   - Amazon Simple Queue Service 佇列
   - Amazon Simple Notification Service 主題
   - AWS Step Functions 狀態機器
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示 Orchestrate Lambda 函數範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

佈建並部署所有資源之後，會顯示 [開始執行] 對話方塊。

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. 「Step Functions」主控台會將您引導至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

關於狀態機及其執行

此範例專案中的狀態機器 AWS Lambda 透過將參數直接傳遞至這些資源與整合，使用 Amazon SQS 佇列管理人工核准請求，並使用 Amazon SNS 主題傳回查詢結果。

Step Functions 執行會接收 JSON 文字做為輸入，並將該輸入傳遞至工作流程中的第一個狀態。各州接收 JSON 數據作為輸入，通常將 JSON 數據作為輸出傳遞到下一個狀態。在此範例專案中，每個步驟的輸出都會作為輸入傳遞至工作流程中的下一個步驟。例如，「產生買入/賣出」建議步驟會接收「檢查股票價格」步驟的輸出作為輸入。此外，「產生買/賣」建議步驟的輸出會作為輸入傳遞至下一個步驟「請求人工核准」，該步驟會模仿人工核准步驟。

> **Note**
>
> 若要檢視步驟傳回的輸出以及傳遞至步驟的輸入，請開啟工作流程執行的「執行詳細資訊」頁面。在「步驟詳細資訊」區段中，檢視您在「檢視」模式中選取之每個步驟的輸入和輸出。

若要實作人工核准步驟，您通常會暫停工作流程執行，直到傳回工作 Token 為止。在此範例專案中，訊息會傳遞至 Amazon SQS 佇列，該佇列用作定義用來處理回呼功能之 Lambda 函數的觸發程序。訊息包含工作 Token，以及前述步驟傳回的輸出。Lambda 函數會使用訊息的有效負載來叫用。工作流程執行會暫停，直到它收到具有 SendTaskSuccessAPI 呼叫的工作 Token 為止。如需有關工作權杖的更多資訊，請參閱等候傳回任務字符的回呼。

此StepFunctionsSample-HelloLambda-ApproveSqsLambda函數的下列程式碼顯示如何將其定義為自動核准 Amazon SQS 佇列透過「Step Functions 數」狀態機提交的任何任務。

用於處理回呼功能並傳回任務權杖的 Lambda 函數程式碼範例

```
exports.lambdaHandler = (event, context, callback) => {
  const stepfunctions = new aws.StepFunctions();

  // For every record in sqs queue
  for (const record of event.Records) {
      const messageBody = JSON.parse(record.body);
      const taskToken = messageBody.TaskToken;

      const params = {
          output: "\"approved\"",
          taskToken: taskToken
      };

      console.log(`Calling Step Functions to complete callback task with params
 ${JSON.stringify(params)}`);

      // Approve
      stepfunctions.sendTaskSuccess(params, (err, data) => {
          if (err) {
              console.error(err.message);
              callback(err.message);
              return;
          }
          console.log(data);
          callback(null);
      });
  }
};
```

瀏覽此範例狀態機器，瞭解 Step Functions 如何控制 Lambda 和 Amazon SQS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "StartAt": "Check Stock Price",
    "States": {
        "Check Stock Price": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLam-
CheckStockPriceLambda-444455556666",
            "Next": "Generate Buy/Sell recommendation"
        },
        "Generate Buy/Sell recommendation": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-Hello-
GenerateBuySellRecommend-123456789012",
            "ResultPath": "$.recommended_type",
            "Next": "Request Human Approval"
        },
        "Request Human Approval": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
            "Parameters": {
                "QueueUrl": "https://sqs.us-west-1.amazonaws.com/111122223333/
StepFunctionsSample-HelloLambda4444-5555-6666-RequestHumanApprovalSqs-777788889999",
                "MessageBody": {
                    "Input.$": "$",
                    "TaskToken.$": "$$.Task.Token"
                }
            },
            "ResultPath": null,
            "Next": "Buy or Sell?"
        },
        "Buy or Sell?": {
            "Type": "Choice",
            "Choices": [
                {
                    "Variable": "$.recommended_type",
                    "StringEquals": "buy",
                    "Next": "Buy Stock"
                },
```

```
                {
                    "Variable": "$.recommended_type",
                    "StringEquals": "sell",
                    "Next": "Sell Stock"
                }
            ]
        },
        "Buy Stock": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
BuyStockLambda-000000000000",
            "Next": "Report Result"
        },
        "Sell Stock": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
SellStockLambda-111111111111",
            "Next": "Report Result"
        },
        "Report Result": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sns:publish",
            "Parameters": {
                "TopicArn": "arn:aws:sns:us-west-1:111122223333:StepFunctionsSample-
HelloLambda1111-2222-3333-ReportResultSnsTopic-222222222222",
                "Message": {
                    "Input.$": "$"
                }
            },
            "End": true
        }
    }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLam-
CheckStockPriceLambda-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-Hello-
GenerateBuySellRecommend-123456789012",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
BuyStockLambda-777788889999",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
SellStockLambda-000000000000",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage*"
            ],
            "Resource": "arn:aws:sqs:us-west-1:111122223333:StepFunctionsSample-
HelloLambda4444-5555-6666-RequestHumanApprovalSqs-111111111111",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-1:111122223333:StepFunctionsSample-
HelloLambda1111-2222-3333-ReportResultSnsTopic-222222222222",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

開始 Athena 查詢

此範例專案以標準工作流程為基礎，示範如何使用 Step Functions 和 Amazon Athena 啟動 Athena 查詢，並傳送包含查詢結果的通知。

在此專案中，Step Functions 數會使用 Lambda 函數和 AWS Glue 爬蟲程式來產生一組範例資料。然後，它會使用 Athena 服務整合執行查詢，並使用 SNS 主題傳回結果。

如需有關 Athena 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 使用 Step Functions 呼叫 Athena

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱 Athena 定價。

步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Start an Athena query`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [啟動Athena查詢]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 一個Amazon Athena查詢
   - 一個 AWS Glue 編目程式

- Amazon SNS 主題
- AWS Step Functions 狀態機器
- 相關 AWS Identity and Access Management (IAM) 角色

下列影像顯示 [啟動Athena查詢範例專案] 的工作流程圖形：

5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀

態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器與 Athena 整合，並 AWS Lambda 將參數直接傳遞至這些資源，並使用 SNS 主題傳回查詢結果。

瀏覽此範例狀態機器，瞭解 Step Functions 如何控制 Lambda 和 Athena。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "StartAt": "Generate example log",
  "States": {
    "Generate example log": {
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athena-LambdaForDataGeneration-AKIAIOSFODNN7EXAMPLE",
      "Type": "Task",
```

```
      "Next": "Run Glue crawler"
    },
    "Run Glue crawler": {
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athen-LambdaForInvokingCrawler-AKIAI44QH8DHBEXAMPLE",
      "Type": "Task",
      "Next": "Start an Athena query"
    },
    "Start an Athena query": {
      "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
      "Parameters": {
        "QueryString": "SELECT * FROM \"athena-sample-project-db-wJalrXUtnFEMI\".\"log
\" limit 1",
        "WorkGroup": "stepfunctions-athena-sample-project-workgroup-wJalrXUtnFEMI"
      },
      "Type": "Task",
      "Next": "Get query results"
    },
    "Get query results": {
      "Resource": "arn:aws:states:::athena:getQueryResults",
      "Parameters": {
        "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
      },
      "Type": "Task",
      "Next": "Send query results"
    },
    "Send query results": {
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "TopicArn": "arn:aws:sns:us-east-1:111122223333:StepFunctionsSample-
AthenaDataQueryd1111-2222-3333-777788889999-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE",
        "Message": {
          "Input.$": "$.ResultSet.Rows"
        }
      },
      "Type": "Task",
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athena-LambdaForDataGeneration-AKIAIOSFODNN7EXAMPLE",
                "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athen-LambdaForInvokingCrawler-AKIAI44QH8DHBEXAMPLE"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:StepFunctionsSample-
AthenaDataQueryd1111-2222-3333-777788889999-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "athena:getQueryResults",
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:111122223333:workgroup/stepfunctions-athena-
sample-project-workgroup-wJalrXUtnFEMI",
                "arn:aws:athena:us-east-1:111122223333:datacatalog/*"
            ],
```

```
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:database/*",
                "arn:aws:glue:us-east-1:111122223333:table/*",
                "arn:aws:glue:us-east-1:111122223333:catalog"
            ],
            "Effect": "Allow"
```

```
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

執行多個查詢（Amazon Athena，Amazon SNS）

此範例專案示範如何連續執行 Athena 查詢，然後 parallel 處理錯誤，然後根據查詢成功還是失敗傳送 Amazon SNS 通知。

在這個專案中，Step Functions 會使用狀態機器同步執行 Athena 查詢。傳回查詢結果後，請輸入 parallel 狀態，並同時執行兩個 Athena 查詢。然後，它會等待任務成功或失敗，並傳送 Amazon SNS 主題，其中包含任務是成功還是失敗的訊息。

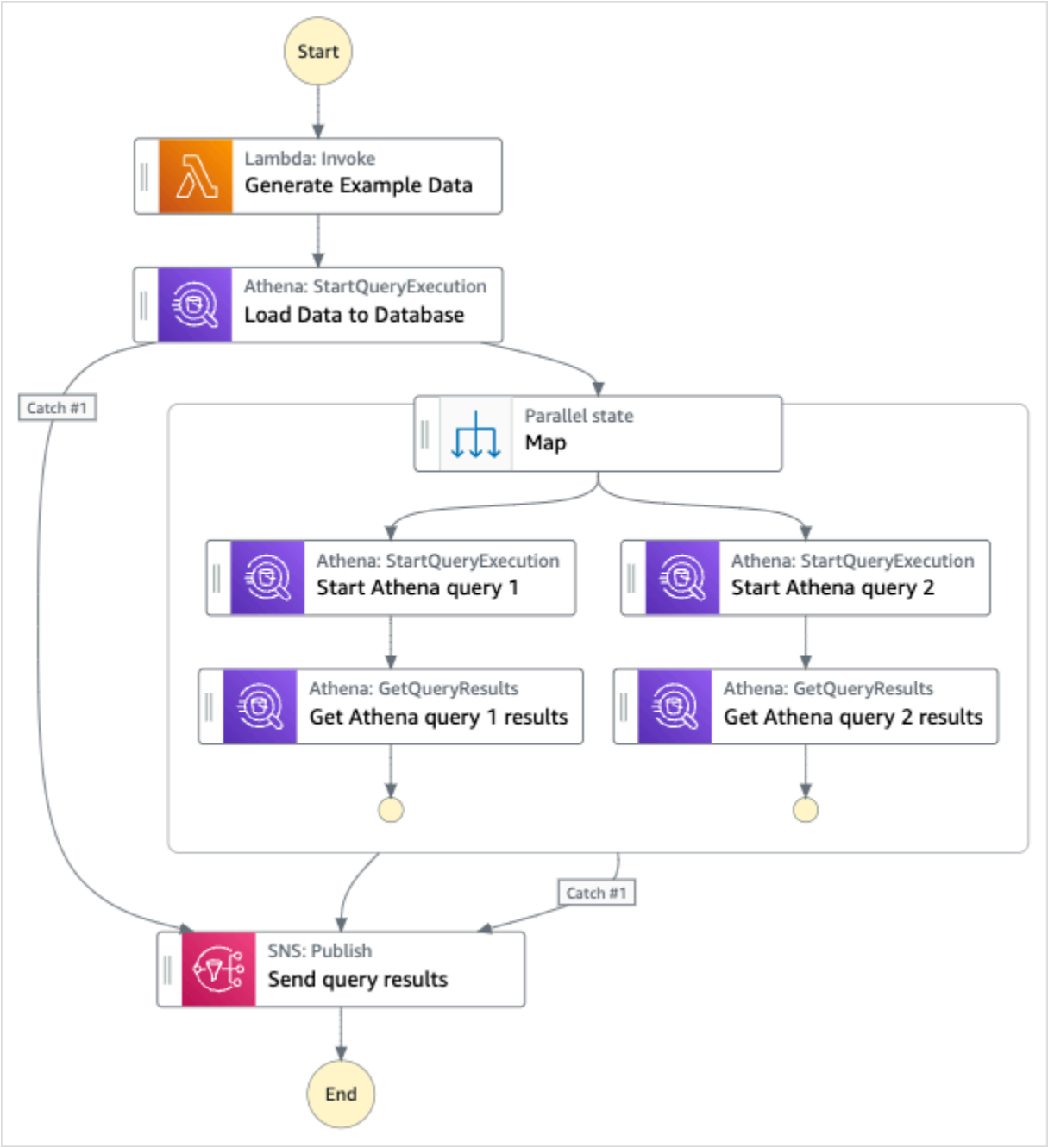
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Execute multiple queries`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [執行多個查詢]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - Amazon Athena查詢
   - Amazon SNS 主題
   - AWS Step Functions 狀態機器
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖展示了「執行多個查詢」範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器

的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 Amazon Athena 和 Amazon SNS 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN) 並傳遞`Parameters`至服務 API 來控制 Amazon Athena 和 Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "Comment": "An example of using Athena to execute queries in sequence and parallel,
 with error handling and notifications.",
    "StartAt": "Generate Example Data",
    "States": {
      "Generate Example Data": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "OutputPath": "$.Payload",
        "Parameters": {
          "FunctionName": "<ATHENA_FUNCTION_NAME>"
        },
```

```
        "Next": "Load Data to Database"
      },
      "Load Data to Database": {
        "Type": "Task",
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
          "QueryString": "<ATHENA_QUERYSTRING>",
          "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Catch": [
          {
            "ErrorEquals": [
              "States.ALL"
            ],
            "Next": "Send query results"
          }
        ],
        "Next": "Map"
      },
      "Map": {
        "Type": "Parallel",
        "ResultSelector": {
          "Query1Result.$": "$[0].ResultSet.Rows",
          "Query2Result.$": "$[1].ResultSet.Rows"
        },
        "Catch": [
          {
            "ErrorEquals": [
              "States.ALL"
            ],
            "Next": "Send query results"
          }
        ],
        "Branches": [
          {
            "StartAt": "Start Athena query 1",
            "States": {
              "Start Athena query 1": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
                "Parameters": {
                  "QueryString": "<ATHENA_QUERYSTRING>",
                  "WorkGroup": "<ATHENA_WORKGROUP>"
                },
```

```
            "Next": "Get Athena query 1 results"
          },
          "Get Athena query 1 results": {
            "Type": "Task",
            "Resource": "arn:aws:states:::athena:getQueryResults",
            "Parameters": {
              "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
            },
            "End": true
          }
        }
      },
      {
        "StartAt": "Start Athena query 2",
        "States": {
          "Start Athena query 2": {
            "Type": "Task",
            "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
            "Parameters": {
                "QueryString": "<ATHENA_QUERYSTRING>",
                "WorkGroup": "<ATHENA_WORKGROUP>"
            },
            "Next": "Get Athena query 2 results"
          },
          "Get Athena query 2 results": {
            "Type": "Task",
            "Resource": "arn:aws:states:::athena:getQueryResults",
            "Parameters": {
              "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
            },
            "End": true
          }
        }
      }
    ],
    "Next": "Send query results"
  },
  "Send query results": {
    "Type": "Task",
    "Resource": "arn:aws:states:::sns:publish",
    "Parameters": {
      "Message.$": "$",
      "TopicArn": "<SNS_TOPIC_ARN>"
    },
```

```
        "End": true
      }
    }
  }
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

AthenaStartQueryExecution

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-ztuvu9yuix",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-2:123456789012:catalog",
                "arn:aws:glue:us-east-2:123456789012:database/*",
                "arn:aws:glue:us-east-2:123456789012:table/*",
                "arn:aws:glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AthenaGetQueryResults

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:getQueryResults"
            ],
            "Resource": [
                "arn:aws:us-east-2:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        }
    ]
}
```

短吻英文

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-
AthenaMultipleQueriese1ec229b-5cbe-4754-a8a8-078474bac878-SNSTopic-9AID0HEJT7TH"
            ]
```

```
        }
    ]
}
```

LambdaInvokeFunction

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-2:123456789012:function:StepFunctionsSample-
Athen-LambdaForStringGeneratio-GQFQjN7mE9gl:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-2:123456789012:function:StepFunctionsSample-
Athen-LambdaForStringGeneratio-GQFQjN7mE9gl"
            ]
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

查詢大型資料集 (Amazon Athena、Amazon S3 AWS Glue、Amazon SNS)

此範例專案示範如何擷取 Amazon S3 中的大型資料集，並透過 AWS Glue 檢索器對其進行分割，然後對該分區執行 Amazon Athena 查詢。

在此專案中，Step Functions 狀態機器會叫用在 Amazon S3 中分割大型資料集的 AWS Glue 搜尋器。一旦 AWS Glue 爬行者程式傳回成功訊息，工作流程就會針對該分割區執行 Athena 查詢。一旦查詢執行成功完成，Amazon SNS 通知便會傳送至 Amazon SNS 主題。

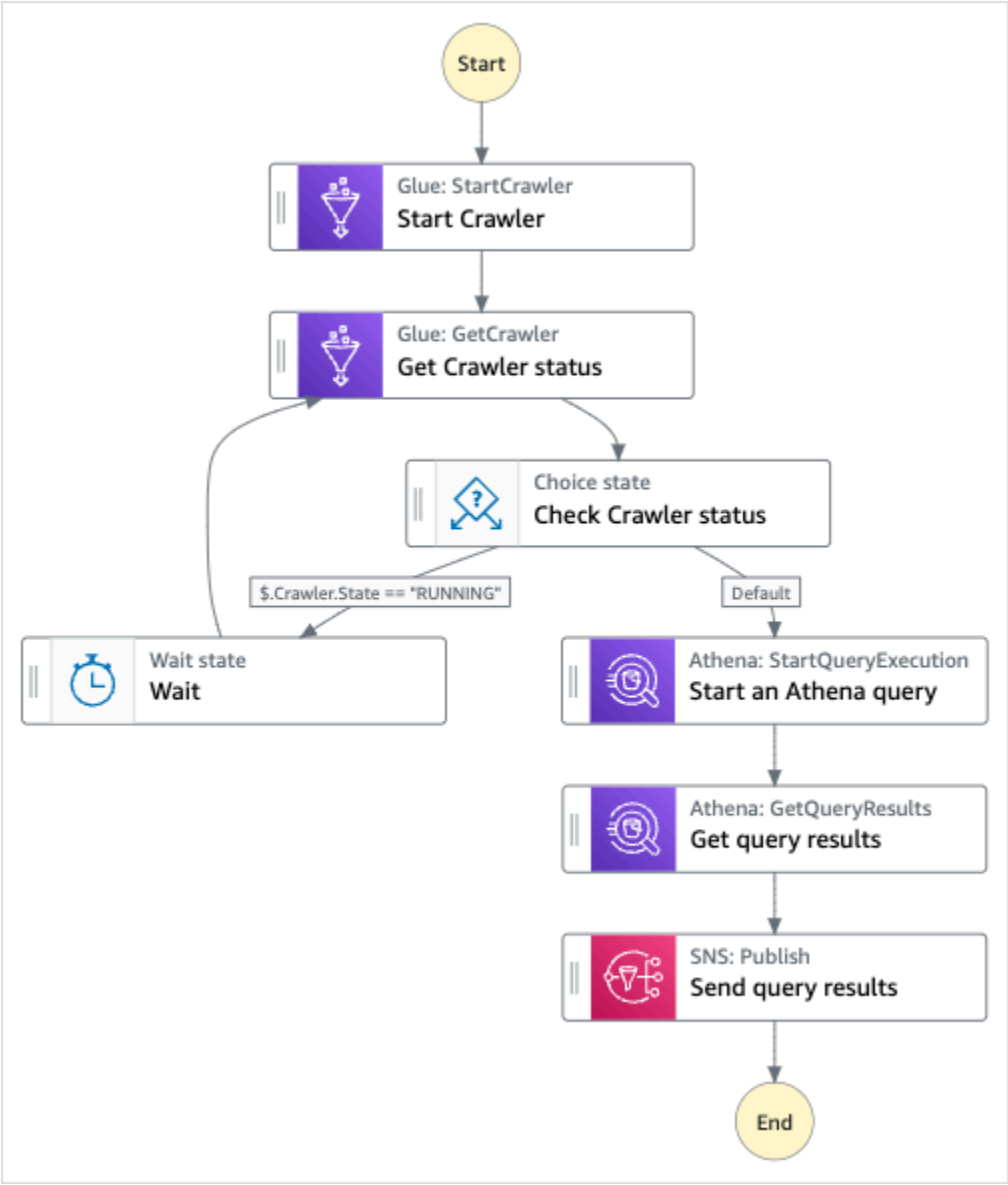
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Query large datasets`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [查詢大型資料集]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - Amazon S3 儲存貯體
   - 一個 AWS Glue 編目程式
   - Amazon SNS 主題
   - AWS Step Functions 狀態機器
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示了「查詢大型資料集」範例專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源 AWS Glue，與 Amazon S3、亞馬遜 Amazon Athena 和 Amazon SNS 整合。

瀏覽此範例狀態機器， AWS Glue瞭解 Step Functions 如何透過連線至現`Resource`場的 Amazon 資源名稱 (ARN) 並傳遞`Parameters`至服務 API 來控制 Amazon S3、亞馬遜 Amazon Athena 和 Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "Comment": "An example demonstrates how to ingest a large data set in Amazon S3 and
 partition it through aws Glue Crawlers, then execute Amazon Athena queries against
 that partition.",
   "StartAt": "Start Crawler",
   "States": {
     "Start Crawler": {
       "Type": "Task",
       "Next": "Get Crawler status",
       "Parameters": {
```

```
          "Name": "<GLUE_CRAWLER_NAME>"
        },
        "Resource": "arn:aws:states:::aws-sdk:glue:startCrawler"
      },
      "Get Crawler status": {
        "Type": "Task",
        "Parameters": {
          "Name": "<GLUE_CRAWLER_NAME>"
        },
        "Resource": "arn:aws:arn:aws:states:::aws-sdk:glue:getCrawler",
        "Next": "Check Crawler status"
      },
      "Check Crawler status": {
        "Type": "Choice",
        "Choices": [
          {
            "Variable": "$.Crawler.State",
            "StringEquals": "RUNNING",
            "Next": "Wait"
          }
        ],
        "Default": "Start an Athena query"
      },
      "Wait": {
        "Type": "Wait",
        "Seconds": 30,
        "Next": "Get Crawler status"
      },
      "Start an Athena query": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "Next": "Get query results"
      },
      "Get query results": {
        "Resource": "arn:aws:states:::athena:getQueryResults",
        "Parameters": {
          "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
        },
        "Type": "Task",
        "Next": "Send query results"
```

```
      },
      "Send query results": {
        "Resource": "arn:aws:states:::sns:publish",
        "Parameters": {
          "TopicArn": "<SNS_TOPIC_ARN>",
          "Message": {
            "Input.$": "$.ResultSet.Rows"
          }
        },
        "Type": "Task",
        "End": true
      }
    }
  }
```

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

AthenaGetQueryResults

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:getQueryResults"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
```

```
        }
    ]
}
```

AthenaStartQueryExecution

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-8v7bshiv70",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
```

```
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-2:123456789012:catalog",
                "arn:aws:glue:us-east-2:123456789012:database/*",
                "arn:aws:glue:us-east-2:123456789012:table/*",
                "arn:aws:glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

短吻英文

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-
AthenaIngestLargeDataset92bc4949-abf8-4a1e-9236-5b7c81b3efa3-SNSTopic-8Y5ZLI5AASXV"
            ]
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

讓資料保持最新狀態 (Amazon Athena、Amazon S3 AWS Glue)

此範例專案示範如何使用 AWS Glue Catalog 查詢目標資料表以取得目前資料，然後使用 Amazon Athena 使用其他來源的新資料更新資料。

在此專案中，Step Functions 狀態機器會呼叫目 AWS Glue 錄，以驗證 Amazon S3 儲存貯體中是否存在目標資料表。如果沒有找到一個表，它將創建一個新表。然後，Step Functions執行 Athena 查詢，將資料列從不同的資料來源新增至目標資料表：先查詢目標資料表以取得最新的日期，然後查詢來源資料表是否有最新的資料，然後將其插入目標資料表。

步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **Keep data up to date**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [讓資料保持為最新狀態]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

- Amazon S3 儲存貯體
- Amazon Athena查詢
- 一個AWS Glue Data Catalog電話
- AWS Step Functions 狀態機器
- 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示了「保持資料為最新」範例專案的工作流程圖形：

5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源 AWS Glue，與 Amazon S3 和 Amazon Athena 整合。

瀏覽此範例狀態機器， AWS Glue瞭解 Step Functions 如何透過連線至現`Resource`場的 Amazon 資源名稱 (ARN) 並傳遞至服務 API `Parameters` 來控制 Amazon S3 和亞馬遜雅典娜。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
    "Comment": "An example demonstrates how to use Athena to query a target table to
 get current data, then update it with new data from other sources.",
    "StartAt": "Get Target Table",
    "States": {
      "Get Target Table": {
        "Type": "Task",
        "Parameters": {
          "DatabaseName": "<GLUE_DATABASE_NAME>",
          "Name": "target"
        },
```

```
        "Catch": [
          {
            "ErrorEquals": [
              "Glue.EntityNotFoundException"
            ],
            "Next": "Create Target Table"
          }
        ],
        "Resource": "arn:aws:states:::aws-sdk:glue:getTable",
        "Next": "Update Target Table"
      },
      "Create Target Table": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "Next": "Update Target Table"
      },
      "Update Target Table": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "End": true
      }
    }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

AthenaStartQueryExecution

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

```
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-26ujlyawxg",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
```

```
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws::glue:us-east-2:123456789012:catalog",
                "arn:aws::glue:us-east-2:123456789012:database/*",
                "arn:aws::glue:us-east-2:123456789012:table/*",
                "arn:aws::glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

管理 Amazon EKS 叢集

此範例專案示範如何使用 Step Functions 和 Amazon Elastic Kubernetes Service 建立具有節點群組的 Amazon EKS 叢集、在 Amazon EKS 上執行任務，然後檢查輸出。完成後，它會移除節點群組和 Amazon EKS 叢集。

如需有關步驟函數和步驟函數服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 使用 Step Functions 調用 Amazon EKS

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱 Amazon EKS 定價。

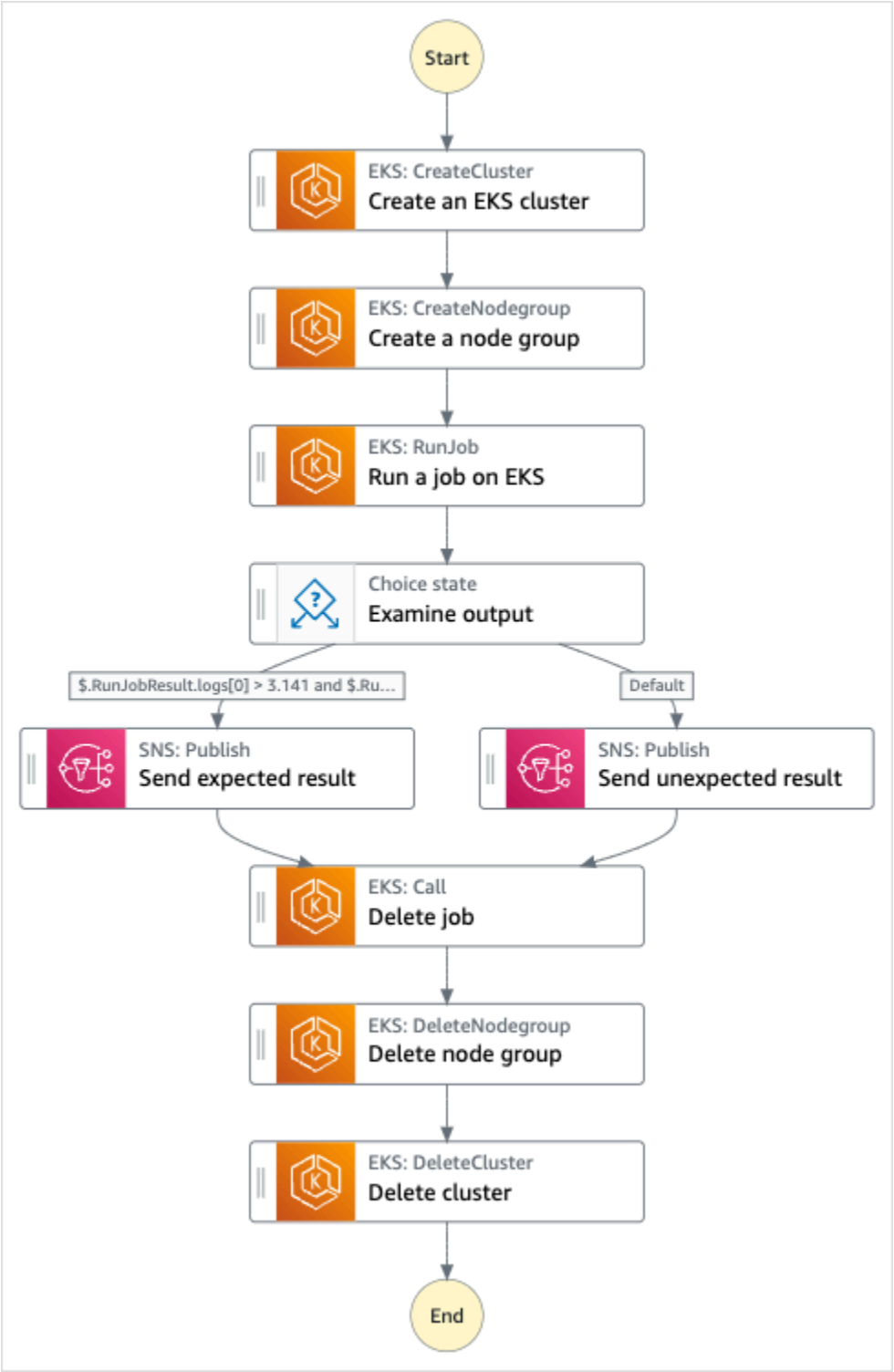
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Manage an EKS cluster`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇 [管理 EKS 叢集]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - Amazon Elastic Kubernetes Service叢集
   - Amazon SNS 主題
   - AWS Step Functions 狀態機器
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示「管理 EKS 叢集範例專案」的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機會透過建立 Amazon EKS 叢集和節點群組與 Amazon EKS 整合，並使用 SNS 主題傳回結果。

瀏覽此範例狀態機器，了解 Step Functions 如何管理 Amazon EKS 叢集和節點群組。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for running Amazon EKS Cluster",
  "StartAt": "Create an EKS cluster",
  "States": {
    "Create an EKS cluster": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createCluster.sync",
```

```
      "Parameters": {
        "Name": "ExampleCluster",
        "ResourcesVpcConfig": {
          "SubnetIds": [
            "subnet-0aacf887d9f00e6a7",
            "subnet-0e5fc41e7507194ab"
          ]
        },
        "RoleArn": "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
      },
      "Retry": [{
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 30,
        "MaxAttempts": 2,
        "BackoffRate": 2
      }],
      "ResultPath": "$.eks",
      "Next": "Create a node group"
    },
    "Create a node group": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createNodegroup.sync",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "NodegroupName": "ExampleNodegroup",
        "NodeRole": "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterMan-
NodeInstanceRole-ANPAJ2UCCR6DPCEXAMPLE",
        "Subnets": [
          "subnet-0aacf887d9f00e6a7",
          "subnet-0e5fc41e7507194ab"]
      },
      "Retry": [{
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 30,
        "MaxAttempts": 2,
        "BackoffRate": 2
      }],
      "ResultPath": "$.nodegroup",
      "Next": "Run a job on EKS"
    },
    "Run a job on EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:runJob.sync",
```

```
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "CertificateAuthority.$": "$.eks.Cluster.CertificateAuthority.Data",
        "Endpoint.$": "$.eks.Cluster.Endpoint",
        "LogOptions": {
          "RetrieveLogs": true
        },
        "Job": {
          "apiVersion": "batch/v1",
          "kind": "Job",
          "metadata": {
            "name": "example-job"
          },
          "spec": {
            "backoffLimit": 0,
            "template": {
              "metadata": {
                "name": "example-job"
              },
              "spec": {
                "containers": [
                  {
                    "name": "pi-20",
                    "image": "perl",
                    "command": [
                      "perl"
                    ],
                    "args": [
                      "-Mbignum=bpi",
                      "-wle",
                      "print '{ ' . '\"pi\": '. bpi(20) . ' }';"
                    ]
                  }
                ],
                "restartPolicy": "Never"
              }
            }
          }
        }
      },
      "ResultSelector": {
        "status.$": "$.status",
        "logs.$": "$.logs..pi"
      },
```

```
      "ResultPath": "$.RunJobResult",
      "Next": "Examine output"
    },
    "Examine output": {
      "Type": "Choice",
      "Choices": [
        {
          "And": [
            {
              "Variable": "$.RunJobResult.logs[0]",
              "NumericGreaterThan": 3.141
            },
            {
              "Variable": "$.RunJobResult.logs[0]",
              "NumericLessThan": 3.142
            }
          ],
          "Next": "Send expected result"
        }
      ],
      "Default": "Send unexpected result"
    },
    "Send expected result": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "TopicArn": "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE",
        "Message": {
          "Input.$": "States.Format('Saw expected value for pi: {}',
 $.RunJobResult.logs[0])"
        }
      },
      "ResultPath": "$.SNSResult",
      "Next": "Delete job"
    },
    "Send unexpected result": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "TopicArn": "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE",
        "Message": {
```

```
          "Input.$": "States.Format('Saw unexpected value for pi: {}',
 $.RunJobResult.logs[0])"
        }
      },
      "ResultPath": "$.SNSResult",
      "Next": "Delete job"
    },
    "Delete job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "CertificateAuthority.$": "$.eks.Cluster.CertificateAuthority.Data",
        "Endpoint.$": "$.eks.Cluster.Endpoint",
        "Method": "DELETE",
        "Path": "/apis/batch/v1/namespaces/default/jobs/example-job"
      },
      "ResultSelector": {
        "status.$": "$.ResponseBody.status"
      },
      "ResultPath": "$.DeleteJobResult",
      "Next": "Delete node group"
    },
    "Delete node group": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "ExampleCluster",
        "NodegroupName": "ExampleNodegroup"
      },
      "Next": "Delete cluster"
    },
    "Delete cluster": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "ExampleCluster"
      },
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:CreateCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:DeleteCluster"
            ],
            "Resource": "arn:aws:eks:sa-east-1:111122223333:cluster/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE"
            ]
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

呼叫 API Gateway

此範例專案示範如何使用 Step Functions 式呼叫 API Gateway，並檢查呼叫是否成功。

如需 API Gateway 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 使用 Step Functions 呼叫 API Gateway

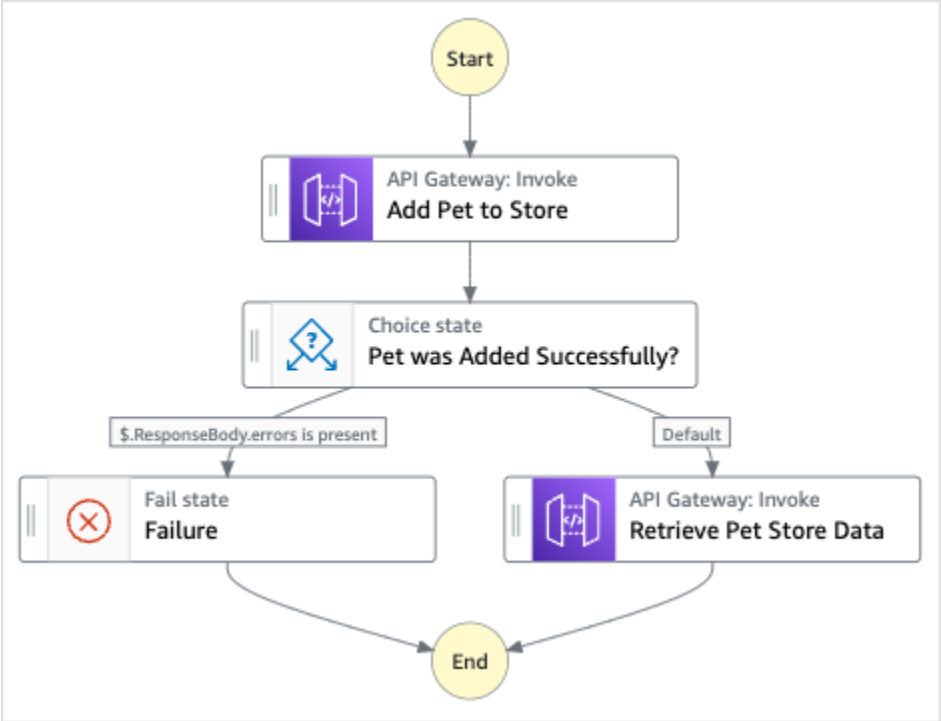
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **Make a call to API Gateway**在搜尋方塊中輸入，然後API Gateway從傳回的搜尋結果中選擇 [撥打電話給]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

- 由狀態機器調用的 Amazon API Gateway REST API。
- AWS Step Functions 狀態機器
- 相關 AWS Identity and Access Management (IAM) 角色

下圖顯示撥打API Gateway樣本專案的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

- 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

  在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會呼叫 API Gateway REST API 並傳遞任何必要的參數，與 API Gateway 整合。

瀏覽此示例狀態機器，以了解 Step Functions 如何與 API Gateway 進行交互。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "Calling APIGW REST Endpoint",
  "StartAt": "Add Pet to Store",
  "States": {
    "Add Pet to Store": {
      "Type": "Task",
      "Resource": "arn:aws:states:::apigateway:invoke",
```

```
      "Parameters": {
        "ApiEndpoint": "<POST_PETS_API_ENDPOINT>",
        "Method": "POST",
        "Stage": "default",
        "Path": "pets",
        "RequestBody.$": "$.NewPet",
        "AuthType": "IAM_ROLE"
      },
      "ResultSelector": {
        "ResponseBody.$": "$.ResponseBody"
      },
      "Next": "Pet was Added Successfully?"
    },
    "Pet was Added Successfully?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.ResponseBody.errors",
          "IsPresent": true,
          "Next": "Failure"
        }
      ],
      "Default": "Retrieve Pet Store Data"
    },
    "Failure": {
      "Type": "Fail"
    },
    "Retrieve Pet Store Data": {
      "Type": "Task",
      "Resource": "arn:aws:states:::apigateway:invoke",
      "Parameters": {
        "ApiEndpoint": "<GET_PETS_API_ENDPOINT>",
        "Method": "GET",
        "Stage": "default",
        "Path": "pets",
        "AuthType": "IAM_ROLE"
      },
      "ResultSelector": {
        "Pets.$": "$.ResponseBody"
      },
      "ResultPath": "$.ExistingPets",
      "End": true
    }
  }
```

```
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-west-1:111122223333:c8hqe4kdg5/default/GET/
pets",
                "arn:aws:execute-api:us-west-1:111122223333:c8hqe4kdg5/default/POST/
pets"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

使用 API Gateway 整合呼叫在 Fargate 上執行的微服務

此範例專案示範如何使用 Step Functions 呼叫 API Gateway，以便與上的服務互動 AWS Fargate，以及檢查呼叫是否成功。

如需 API Gateway 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 使用 Step Functions 呼叫 API Gateway

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱定價。

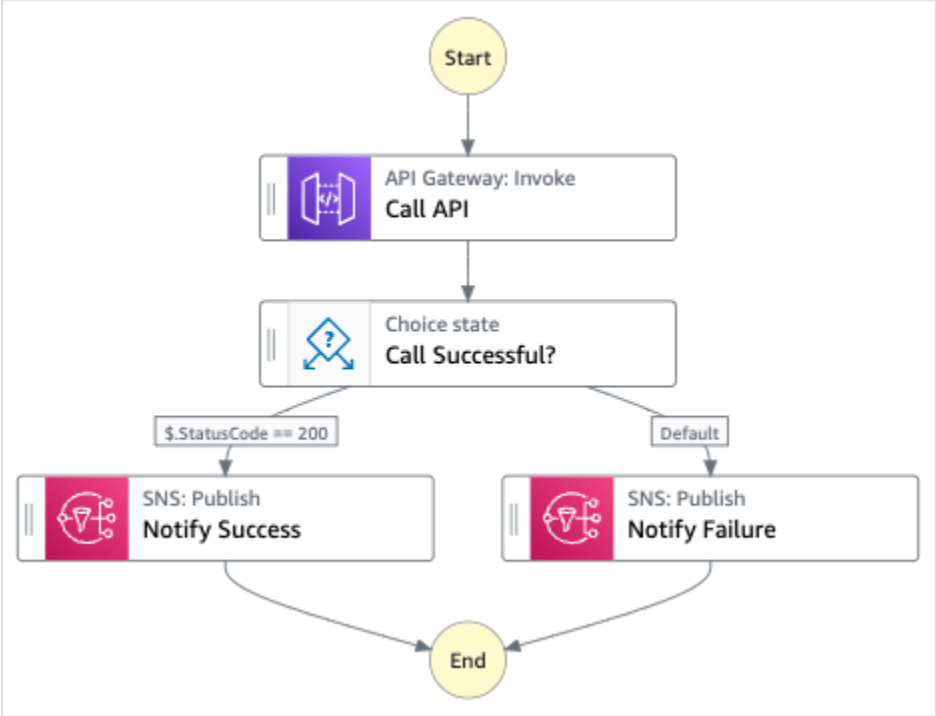
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Call a microservice with API Gateway`**在搜尋方塊中輸入，然後從傳回的搜尋結果API Gateway中選擇呼叫微服務。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 由狀態機器呼叫的 Amazon API Gateway HTTP API。
   - 一個Amazon API GatewayAmazon VPC鏈接。
   - Amazon Virtual Private Cloud。
   - Application Load Balancer。
   - 叢Fargate集。
   - Amazon SNS 主題
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色
   - 若要讓這些資源共同運作，所需的其他幾項服務。

   下圖顯示使用API Gateway範例專案呼叫微服務的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機會呼叫連接至 Fargate 上服務的 API Gateway HTTP API，與 API Gateway 整合。它託管在私有子網路上，並透過私有應用程式負載平衡器存取。

瀏覽此範例狀態機器，以瞭解 Step Functions 式如何與 API Gateway 互動並傳回結果。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "Calling APIGW HTTP Endpoint",
  "StartAt": "Call API",
  "States": {
    "Call API": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::apigateway:invoke",
      "Parameters": {
        "ApiEndpoint": "<API_ENDPOINT>",
        "Method": "GET",
        "AuthType": "IAM_ROLE"
      },
      "Next": "Call Successful?"
    },
    "Call Successful?": {
      "Type": "Choice",
      "Choices": [
        {
```

```
          "Variable": "$.StatusCode",
          "NumericEquals": 200,
          "Next": "Notify Success"
        }
      ],
      "Default": "Notify Failure"
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::sns:publish",
      "Parameters": {
        "Message": "Call was successful",
        "TopicArn": "<SNS_TOPIC_ARN>"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::sns:publish",
      "Parameters": {
        "Message": "Call was not successful",
        "TopicArn": "<SNS_TOPIC_ARN>"
      },
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
```

```
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:apigw-ecs-sample-2000-
SNSTopic-444455556666"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:444444444444/*/GET/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:Describe*",
                "ec2:DetachNetworkInterface",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:Describe*",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
```

```
        {
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

將自訂事件傳送至 EventBridge

此範例專案示範如何使用 Step Functions 將自訂事件傳送至符合具有多個目標之規則的事件匯流排 (Amazon EventBridge AWS Lambda、Amazon 簡單通知服務、Amazon 簡單佇列服務)。

如需有關步驟函數和步驟函數服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- EventBridge 使用 Step Functions 調用

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱EventBridge 定價。

步驟 1：建立狀態機器並佈建資源

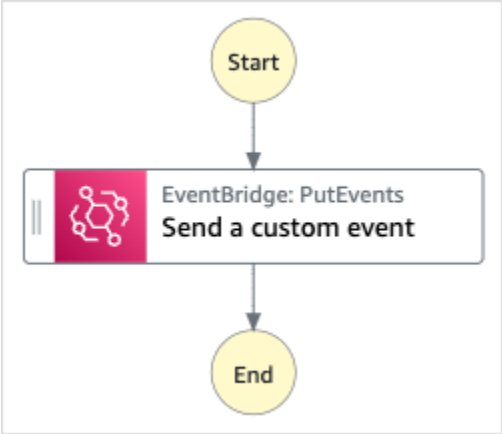
1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。

2. **Send a custom event to EventBridge**在搜尋方塊中輸入，然後EventBridge從傳回的搜尋結果中選擇 [傳送自訂事件至]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇「執行示範」或「在其上建置」。

   此範例專案會部署下列資源：

   - 一個Amazon EventBridge事件
   - Amazon SNS 主題
   - Amazon SQS 佇列
   - Lambda 函數
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下圖顯示「將自訂事件傳送至EventBridge範例專案」的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器

的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. 「Step Functions」主控台會將您引導至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機會透 EventBridge 過將自訂事件傳送至事件匯流排來整合。 EventBridge 傳送至事件匯流排的事件與觸發將訊息傳送至 Amazon SNS 主題和 Amazon SQS 佇列的 Lambda 函數的 EventBridge 規則相符。

瀏覽此範例狀態機器，以查看 Step Functions 如何管理 EventBridge。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for sending a custom event to
 Amazon EventBridge",
  "StartAt": "Send a custom event",
  "States": {
    "Send a custom event": {
```

```
      "Resource": "arn:<PARTITION>:states:::events:putEvents",
      "Type": "Task",
      "Parameters": {
        "Entries": [{
          "Detail": {
            "Message": "Hello from Step Functions!"
          },
          "DetailType": "MessageFromStepFunctions",
          "EventBusName": "<EVENT_BUS_NAME>",
          "Source": "my.statemachine"
        }]
      },
      "End": true
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:1234567890:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

叫用同步快速工作流

此範例專案示範如何透過 Amazon API Gateway 叫用同步快速工作流程來管理員工資料庫。

在此專案中，Step Functions 使用 API Gateway 端點來啟動 Step Functions 同步快速工作流程。然後，這些工具會使用 DynamoDB 來搜尋、新增和移除員工資料庫中的員工。

如需步驟函數同步 Express 工作流程的詳細資訊，請參閱同步和非同步快速工作流。

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱 Step Functions 定價。

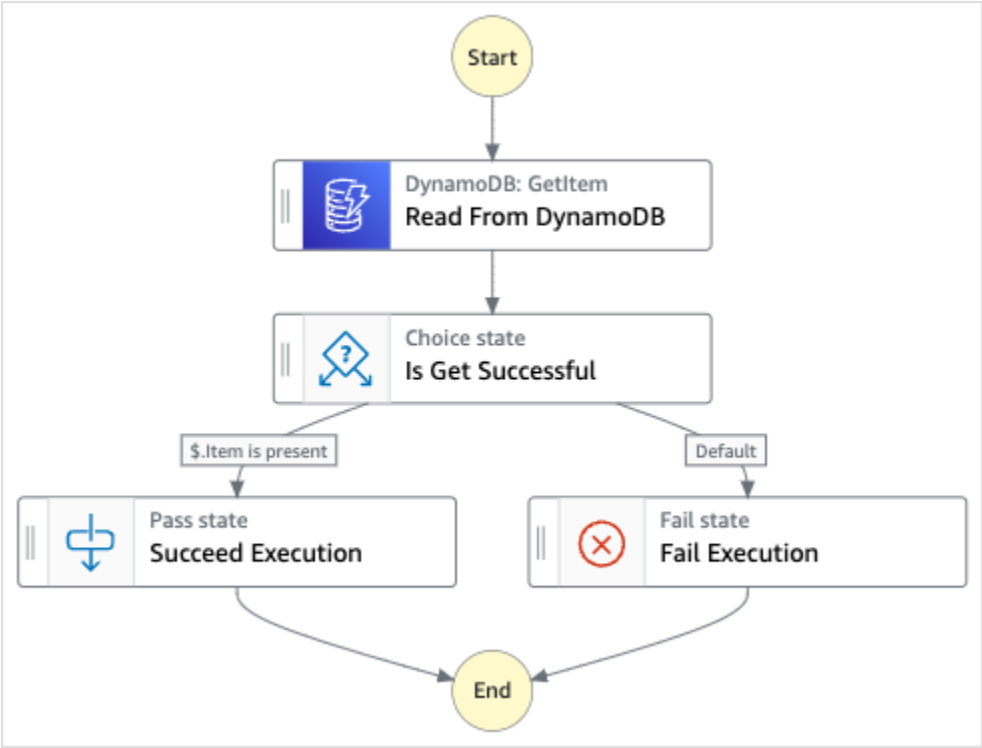
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Invoke Synchronous Express Workflows through API Gateway`**在搜尋方塊中輸入，然後API Gateway從傳回的搜尋結果中選擇 [呼叫同步 Express 工作流程至]。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 由狀態機器呼叫的 Amazon API Gateway HTTPS API。
   - Amazon DynamoDB 資料表。
   - 三個 AWS Step Functions 狀態機。
   - 相關 AWS Identity and Access Management (IAM) 角色。

   下列影像顯示透過API Gateway範例專案叫用同步 Express 工作流程的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

> **Note**
>
> 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器與 API Gateway 和 DynamoDB 整合，方法是使用 API Gateway 叫用同步快速工作流程，然後使用 DynamoDB 從員工資料庫更新或讀取該工作流程。

瀏覽此範例狀態機器，以查看 Step Functions 如何從 DynamoDB 讀取以擷取員工資訊。

若要深入瞭解如何使用 API Gateway 叫用 Step Functions 式，請參閱下列內容。

- 使用 Step Functions 呼叫 API Gateway
- 如何在 API Gateway 開發人員指南中叫用私有閘道。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "This state machine returns an employee entry from DynamoDB",
  "StartAt": "Read From DynamoDB",
  "States": {
    "Read From DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "StepFunctionsSample-
SynchronousExpressWorkflowAKIAIOSFODNN7EXAMPLE-DynamoDBTable-ANPAJ2UCCR6DPCEXAMPLE",
        "Key": {
```

```
          "EmployeeId": {"S.$": "$.employee"}
        }
      },
      "Retry": [
        {
          "ErrorEquals": [
            "DynamoDB.AmazonDynamoDBException"
          ],
          "IntervalSeconds": 3,
          "MaxAttempts": 2,
          "BackoffRate": 1.5
        }
      ],
      "Next": "Is Get Successful"
    },
    "Is Get Successful": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.Item",
          "IsPresent": true,
          "Next": "Succeed Execution"
        }
      ],
      "Default": "Fail Execution"
    },
    "Succeed Execution": {
      "Type": "Pass",
      "Parameters" : {
        "employee.$": "$.Item.EmployeeId.S",
        "jobTitle.$": "$.Item.JobTitle.S"
      },
      "End": true
    },
    "Fail Execution": {
      "Type": "Fail",
      "Error": "EmployeeDoesNotExist"
    }
  }
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:111122223333:table/Write"
            ]
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

使用亞馬遜紅移 (Lambda、亞馬 Amazon Redshift 資料 API) 執行 ETL/ELT 工作流程

此範例專案示範如何使用 Step Functions 數和 Amazon Redshift 資料 API 來執行 ETL/ELT 工作流程，將資料載入 Amazon Redshift 資料倉儲。

在此專案中，Step Functions 會使用 AWS Lambda 函數和 Amazon Redshift 資料 API 來建立所需的資料庫物件並產生一組範例資料，然後 parallel 執行兩個工作來執行載入維度資料表，然後是事實資料表。一旦兩個維度載入任務都順利結束，Step Functions 會為事實資料表執行載入任務、執行驗證任務，然後暫停 Amazon Redshift 叢集。

> **Note**
>
> 您可以修改 ETL 邏輯以接收來自其他來源 (例如 Amazon S3) 的資料，這些來源可以使用 COPY 命令將資料從 Amazon S3 複製到 Amazon Redshift 表。

如需 Amazon Redshift 和 Step Functions 服務整合的詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 使用 Amazon Redshift 數據 API
- Amazon Redshift 數據 API 服務
- 建立使用 Lambda 的 Step Functions 狀態機器
- 以下項目的 IAM 政策：
  - 的 IAM 政策 AWS Lambda
  - 授權存取 Amazon Redshift 資料 API

> **Note**
>
> 此範例專案可能需要付費。
> 對於新用 AWS 戶，可以使用免費用量方案。在此方案中，特定用量層級以下的服務皆為免費。如需有關 AWS 成本和免費方案的詳細資訊，請參閱AWS Step Functions 定價。

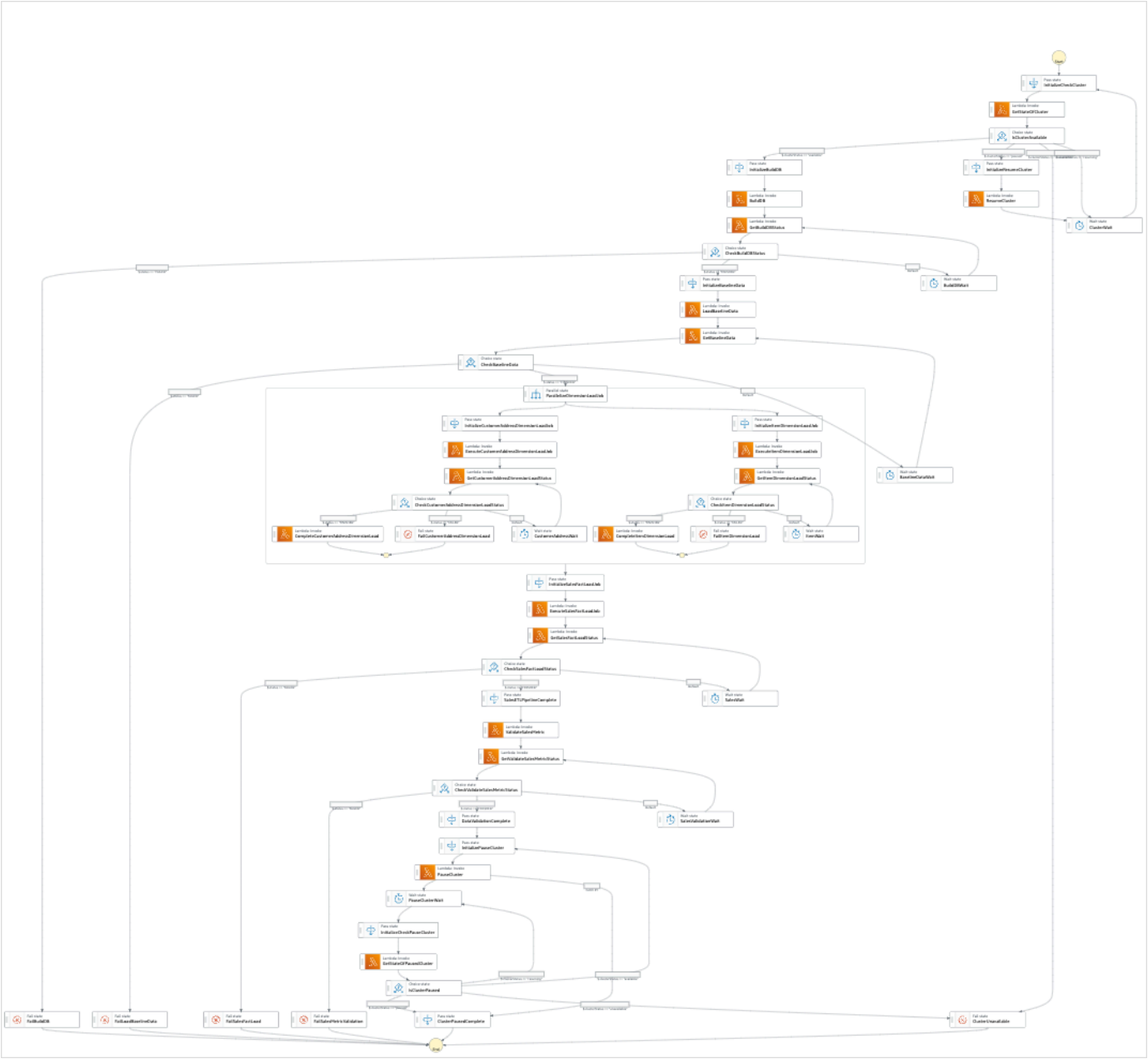
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **ETL job in Amazon Redshift**在搜尋方塊中輸入，然後從傳回的搜尋結果Amazon Redshift中選擇 ETL 工作。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇「執行示範」或「在其上建置」。

   此範例專案會部署下列資源：

   - Amazon Redshift叢集
   - 兩個 Lambda 函數
   - 一個Amazon Redshift架構
   - 五Amazon Redshift張桌子
   - 一个 AWS Step Functions 状态机
   - 相關 AWS Identity and Access Management (IAM) 角色。

   下圖顯示Amazon Redshift範例專案中 ETL 工作的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器 AWS Lambda 透過將 ETL 邏輯作為 InputPath 直接傳遞給這些資源，並使用 Amazon Redshift 資料 API 以非同步方式執行來與之整合。

瀏覽此範例狀態機器，瞭解 Step Functions 數如何控制 AWS Lambda 和 Amazon Redshift 資料 API。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
"Comment": "A simple ETL workflow for loading dimension and fact tables",
  "StartAt": "InitializeCheckCluster",
  "States": {
    "InitializeCheckCluster": {
      "Type": "Pass",
      "Next": "GetStateOfCluster",
```

```
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterAvailable",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterAvailable": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializeBuildDB"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "InitializeResumeCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "ClusterWait"
        }
      ]
    },
    "ClusterWait": {
      "Type": "Wait",
```

```
      "Seconds": 720,
      "Next": "InitializeCheckCluster"
    },
    "InitializeResumeCluster": {
      "Type": "Pass",
      "Next": "ResumeCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "resume"
        }
      }
    },
    "ResumeCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "ClusterWait",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "InitializeBuildDB": {
      "Type": "Pass",
      "Next": "BuildDB",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "build_database",
          "sql_statement": [
            "create schema if not exists {0} authorization {1};",
            "create table if not exists {0}.customer",
            "(c_customer_sk          int4     not null encode az64",
            ",c_customer_id          char(16) not null encode zstd",
            ",c_current_addr_sk      int4              encode az64",
            ",c_first_name           char(20)          encode zstd",
            ",c_last_name            char(30)          encode zstd",
            ",primary key (c_customer_sk)",
            ") distkey(c_customer_sk);",
            "--",
```

```
            "create table if not exists {0}.customer_address",
            "(ca_address_sk    int4     not null encode az64",
            ",ca_address_id    char(16) not null encode zstd",
            ",ca_state         char(2)           encode zstd",
            ",ca_zip           char(10)          encode zstd",
            ",ca_country       varchar(20)       encode zstd",
            ",primary key (ca_address_sk)",
            ") distkey(ca_address_sk);",
            "--",
            "create table if not exists {0}.date_dim",
            "(d_date_sk           integer  not null encode az64",
            ",d_date_id           char(16) not null encode zstd",
            ",d_date              date              encode az64",
            ",d_day_name          char(9)           encode zstd",
            ",primary key (d_date_sk)",
            ") diststyle all;",
            "--",
            "create table if not exists {0}.item",
            "(i_item_sk        int4     not null encode az64",
            ",i_item_id        char(16) not null encode zstd",
            ",i_rec_start_date date              encode az64",
            ",i_rec_end_date   date              encode az64",
            ",i_current_price  numeric(7,2)      encode az64",
            ",i_category       char(50)          encode zstd",
            ",i_product_name   char(50)          encode zstd",
            ",primary key (i_item_sk)",
            ") distkey(i_item_sk) sortkey(i_category);",
            "--",
            "create table if not exists {0}.store_sales",
            "(ss_sold_date_sk       int4",
            ",ss_item_sk            int4 not null encode az64",
            ",ss_customer_sk        int4          encode az64",
            ",ss_addr_sk            int4          encode az64",
            ",ss_store_sk           int4          encode az64",
            ",ss_ticket_number      int8 not null encode az64",
            ",ss_quantity           int4          encode az64",
            ",ss_net_paid           numeric(7,2)  encode az64",
            ",ss_net_profit         numeric(7,2)  encode az64",
            ",primary key (ss_item_sk, ss_ticket_number)",
            ") distkey(ss_item_sk) sortkey(ss_sold_date_sk);"
          ]
        }
      }
    },
```

```
    "BuildDB": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBuildDBStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBuildDBStatus": {
      "Type": "Task",
      "Next": "CheckBuildDBStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBuildDBStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailBuildDB"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "InitializeBaselineData"
        }
      ],
      "Default": "BuildDBWait"
    },
    "BuildDBWait": {
      "Type": "Wait",
      "Seconds": 15,
      "Next": "GetBuildDBStatus"
    },
    "FailBuildDB": {
      "Type": "Fail",
      "Cause": "Database Build Failed",
```

```
      "Error": "Error"
    },
    "InitializeBaselineData": {
      "Type": "Pass",
      "Next": "LoadBaselineData",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "load_baseline_data",
          "sql_statement": [
            "begin transaction;",
            "truncate table {0}.customer;",
            "insert into {0}.customer
(c_customer_sk,c_customer_id,c_current_addr_sk,c_first_name,c_last_name)",
            "values",
            "(7550,'AAAAAAAAOHNBAAAA',9264662,'Michelle','Deaton'),",
            "(37079,'AAAAAAAAHNAJAAAA',13971208,'Michael','Simms'),",
            "(40626,'AAAAAAAACLOJAAAA',1959255,'Susan','Ryder'),",
            "(2142876,'AAAAAAAAMJCLACAA',7644556,'Justin','Brown');",
            "analyze {0}.customer;",
            "--",
            "truncate table {0}.customer_address;",
            "insert into {0}.customer_address
(ca_address_sk,ca_address_id,ca_state,ca_zip,ca_country)",
            "values",
            "(13971208,'AAAAAAAAIAPCFNAA','NE','63451','United States'),",
            "(7644556,'AAAAAAAAMIFKEHAA','SD','58883','United States'),",
            "(9264662,'AAAAAAAAGBOFNIAA','CA','99310','United States');",
            "analyze {0}.customer_address;",
            "--",
            "truncate table {0}.item;",
            "insert into {0}.item
(i_item_sk,i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name
            "values",

"(3417,'AAAAAAAAIFNAAAAA','1997-10-27',NULL,14.29,'Electronics','ationoughtesepri
'),",
            "(9615,'AAAAAAAAOIFCAAAA','1997-10-27',NULL,9.68,'Home','antioughtcallyn
st'),",
            "(3693,'AAAAAAAAMGOAAAAA','2001-03-12',NULL,2.10,'Men','prin
stcallypri'),",
```

```
 "(3630,'AAAAAAAAMCOAAAAA','2001-10-27',NULL,2.95,'Electronics','barpricallypri'),",

 "(16506,'AAAAAAAAIHAEAAAA','2001-10-27',NULL,3.85,'Home','callybaranticallyought'),",

 "(7866,'AAAAAAAAILOBAAAA','2001-10-27',NULL,12.60,'Jewelry','callycallyeingation');",
            "--",
            "analyze {0}.item;",
            "truncate table {0}.date_dim;",
            "insert into {0}.date_dim (d_date_sk,d_date_id,d_date,d_day_name)",
            "values",
            "(2450521,'AAAAAAAAJFEGFCAA','1997-03-13','Thursday'),",
            "(2450749,'AAAAAAAANDFGFCAA','1997-10-27','Monday'),",
            "(2451251,'AAAAAAAADDHGFCAA','1999-03-13','Saturday'),",
            "(2451252,'AAAAAAAAEDHGFCAA','1999-03-14','Sunday'),",
            "(2451981,'AAAAAAAANAKGFCAA','2001-03-12','Monday'),",
            "(2451982,'AAAAAAAAOAKGFCAA','2001-03-13','Tuesday'),",
            "(2452210,'AAAAAAAACPKGFCAA','2001-10-27','Saturday'),",
            "(2452641,'AAAAAAAABKMGFCAA','2003-01-01','Wednesday'),",
            "(2452642,'AAAAAAAACKMGFCAA','2003-01-02','Thursday');",
            "--",
            "analyze {0}.date_dim;",
            "--  commit and End transaction",
            "commit;",
            "end transaction;"
          ]
        }
      }
    },
    "LoadBaselineData": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBaselineData",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBaselineData": {
      "Type": "Task",
      "Next": "CheckBaselineData",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
```

```
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBaselineData": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailLoadBaselineData"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "ParallelizeDimensionLoadJob"
        }
      ],
      "Default": "BaselineDataWait"
    },
    "BaselineDataWait": {
      "Type": "Wait",
      "Seconds": 20,
      "Next": "GetBaselineData"
    },
    "FailLoadBaselineData": {
      "Type": "Fail",
      "Cause": "Load Baseline Data Failed",
      "Error": "Error"
    },
    "ParallelizeDimensionLoadJob": {
      "Type": "Parallel",
      "Next": "InitializeSalesFactLoadJob",
      "ResultPath": "$.status",
      "Branches": [
        {
          "StartAt": "InitializeCustomerAddressDimensionLoadJob",
          "States": {
            "InitializeCustomerAddressDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteCustomerAddressDimensionLoadJob",
              "Result": {
                "input": {
```

```
                "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                "redshift_database": "dev",
                "redshift_user": "awsuser",
                "redshift_schema": "tpcds",
                "action": "load_customer_address",
                "sql_statement": [
                  "begin transaction;",
                  "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                  "drop table if exists {0}.stg_customer_address;",
                  "create table if not exists {0}.stg_customer_address",
                  "(ca_address_id    varchar(16)  encode zstd",
                  ",ca_state         varchar(2)   encode zstd",
                  ",ca_zip           varchar(10)  encode zstd",
                  ",ca_country       varchar(20)  encode zstd",
                  ")",
                  "backup no",
                  "diststyle even;",
                  "/* Ingest data from source */",
                  "insert into {0}.stg_customer_address
(ca_address_id,ca_state,ca_zip,ca_country)",
                  "values",
                  "('AAAAAAAACFBBAAAA','NE','','United States'),",
                  "('AAAAAAAAGAEFAAAA','NE','61749','United States'),",
                  "('AAAAAAAAPJKKAAAA','OK','','United States'),",
                  "('AAAAAAAAMIHGAAAA','AL','','United States');",
                  "/* Perform UPDATE for existing data with refreshed attribute
values */",
                  "update {0}.customer_address",
                  "   set ca_state = stg_customer_address.ca_state,",
                  "       ca_zip = stg_customer_address.ca_zip,",
                  "       ca_country = stg_customer_address.ca_country",
                  "  from {0}.stg_customer_address",
                  " where customer_address.ca_address_id =
stg_customer_address.ca_address_id;",
                  "/* Perform insert for new rows  */",
                  "insert into {0}.customer_address",
                  "(ca_address_sk",
                  ",ca_address_id",
                  ",ca_state",
                  ",ca_zip",
                  ",ca_country",
                  ")",
                  "with max_customer_address_sk as",
```

```
                    "(select max(ca_address_sk) max_ca_address_sk",
                    "from {0}.customer_address)",
                    "select row_number() over (order by
 stg_customer_address.ca_address_id) + max_customer_address_sk.max_ca_address_sk as
 ca_address_sk",
                    ",stg_customer_address.ca_address_id",
                    ",stg_customer_address.ca_state",
                    ",stg_customer_address.ca_zip",
                    ",stg_customer_address.ca_country",
                    "from {0}.stg_customer_address,",
                    "max_customer_address_sk",
                    "where stg_customer_address.ca_address_id not in (select
 customer_address.ca_address_id from {0}.customer_address);",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                  ]
                }
              }
            },
            "ExecuteCustomerAddressDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetCustomerAddressDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetCustomerAddressDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckCustomerAddressDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckCustomerAddressDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
```

```
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailCustomerAddressDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteCustomerAddressDimensionLoad"
                }
              ],
              "Default": "CustomerAddressWait"
            },
            "CustomerAddressWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetCustomerAddressDimensionLoadStatus"
            },
            "CompleteCustomerAddressDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailCustomerAddressDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        },
        {
          "StartAt": "InitializeItemDimensionLoadJob",
          "States": {
            "InitializeItemDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteItemDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
```

```
                  "action": "load_item",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
 created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_item;",
                    "create table if not exists {0}.stg_item",
                    "(i_item_id        varchar(16) encode zstd",
                    ",i_rec_start_date date encode zstd",
                    ",i_rec_end_date   date encode zstd",
                    ",i_current_price  numeric(7,2) encode zstd",
                    ",i_category       varchar(50) encode zstd",
                    ",i_product_name   varchar(50) encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_item",

 "(i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name)",
                    "values",

 "('AAAAAAAAABJBAAAA','2000-10-27',NULL,4.10,'Books','ationoughtesecally'),",
                    "('AAAAAAAAOPKBAAAA','2001-10-27',NULL,4.22,'Books','ableoughtn
 stcally'),",
                    "('AAAAAAAAHGPAAAAA','1997-10-27',NULL,29.30,'Books','priesen
 stpri'),",

 "('AAAAAAAAICMAAAAA','2001-10-27',NULL,1.93,'Books','eseoughtoughtpri'),",

 "('AAAAAAAAGPGBAAAA','2001-10-27',NULL,9.96,'Books','bareingeinganti'),",
                    "('AAAAAAAAANBEBAAAA','1997-10-27',NULL,2.25,'Music','n
 steseoughtanti'),",

 "('AAAAAAAACLAAAAAA','2001-10-27',NULL,1.71,'Home','bareingought'),",

 "('AAAAAAAAOBBDAAAA','2001-10-27',NULL,5.55,'Books','callyationantiableought');",
                    "/
*******************************************************************************
                    "** Type 2 is maintained for i_current_price column.",
                    "** Update all attributes for the item when the price is not
 changed",
                    "** Sunset existing active item record with current i_rec_end_date
 and insert a new record when the price does not match",
```

```
"*************************************************************************************************
                    "update {0}.item",
                    "   set i_category = stg_item.i_category,",
                    "       i_product_name = stg_item.i_product_name",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price = stg_item.i_current_price;",
                    "insert into {0}.item",
                    "(i_item_sk",
                    ",i_item_id",
                    ",i_rec_start_date",
                    ",i_rec_end_date",
                    ",i_current_price",
                    ",i_category",
                    ",i_product_name",
                    ")",
                    "with max_item_sk as",
                    "(select max(i_item_sk) max_item_sk",
                    "   from {0}.item)",
                    "select row_number() over (order by stg_item.i_item_id) +
max_item_sk as i_item_sk",
                    "      ,stg_item.i_item_id",
                    "      ,trunc(sysdate) as i_rec_start_date",
                    "      ,null as i_rec_end_date",
                    "      ,stg_item.i_current_price",
                    "      ,stg_item.i_category",
                    "      ,stg_item.i_product_name",
                    "  from {0}.stg_item, {0}.item, max_item_sk",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Sunset penultimate records that were inserted as type 2 */",
                    "update {0}.item",
                    "   set i_rec_end_date = trunc(sysdate)",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                  ]
```

```
                }
              }
            },
            "ExecuteItemDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetItemDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetItemDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckItemDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckItemDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailItemDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteItemDimensionLoad"
                }
              ],
              "Default": "ItemWait"
            },
            "ItemWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetItemDimensionLoadStatus"
            },
```

```
            "CompleteItemDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailItemDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        }
      ]
    },
    "InitializeSalesFactLoadJob": {
      "Type": "Pass",
      "Next": "ExecuteSalesFactLoadJob",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "load_sales_fact",
          "sql_statement": [
            "begin transaction;",
            "/* Create a stg_store_sales staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "create table {0}.stg_store_sales",
            "(sold_date             date encode zstd",
            ",i_item_id             varchar(16) encode zstd",
            ",c_customer_id         varchar(16) encode zstd",
            ",ca_address_id         varchar(16) encode zstd",
            ",ss_ticket_number      integer encode zstd",
            ",ss_quantity           integer encode zstd",
            ",ss_net_paid           numeric(7,2) encode zstd",
            ",ss_net_profit         numeric(7,2) encode zstd",
            ")",
            "backup no",
            "diststyle even;",
```

```
            "/* Ingest data from source */",
            "insert into {0}.stg_store_sales",

"(sold_date,i_item_id,c_customer_id,ca_address_id,ss_ticket_number,ss_quantity,ss_net_paid,ss_
            "values",

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,5046.37,150

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,2103.72,-12

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,959.10,-130

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1403191,13,962.65,-475

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,111.60,-241

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,4013.02,-11

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',1201746,17,2689.12,-55

"('2003-01-02','AAAAAAAAMGOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',193971,18,1876.89,-556
            "/* Delete any rows from target store_sales for the input date for
idempotency */",
            "delete from {0}.store_sales where ss_sold_date_sk in (select d_date_sk
from {0}.date_dim where d_date='{1}');",
            "/* Insert data from staging table to the target table */",
            "insert into {0}.store_sales",
            "(ss_sold_date_sk",
            ",ss_item_sk",
            ",ss_customer_sk",
            ",ss_addr_sk",
            ",ss_ticket_number",
            ",ss_quantity",
            ",ss_net_paid",
            ",ss_net_profit",
            ")",
            "select date_dim.d_date_sk ss_sold_date_sk",
            "      ,item.i_item_sk ss_item_sk",
            "      ,customer.c_customer_sk ss_customer_sk",
            "      ,customer_address.ca_address_sk ss_addr_sk",
            "      ,ss_ticket_number",
            "      ,ss_quantity",
            "      ,ss_net_paid",
            "      ,ss_net_profit",
```

```
            "  from {0}.stg_store_sales as store_sales",
            "  inner join {0}.date_dim on store_sales.sold_date = date_dim.d_date",
            "  left join {0}.item on store_sales.i_item_id = item.i_item_id and
 item.i_rec_end_date is null",
            "  left join {0}.customer on store_sales.c_customer_id =
 customer.c_customer_id",
            "  left join {0}.customer_address on store_sales.ca_address_id =
 customer_address.ca_address_id;",
            "/* Drop staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "/* Commit and End transaction */",
            "commit;",
            "end transaction;"
          ]
        }
      }
    },
    "ExecuteSalesFactLoadJob": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetSalesFactLoadStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetSalesFactLoadStatus": {
      "Type": "Task",
      "Next": "CheckSalesFactLoadStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckSalesFactLoadStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesFactLoad"
```

```
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "SalesETLPipelineComplete"
        }
      ],
      "Default": "SalesWait"
    },
    "SalesWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetSalesFactLoadStatus"
    },
    "FailSalesFactLoad": {
      "Type": "Fail",
      "Cause": "ETL Workflow Failed",
      "Error": "Error"
    },
    "ClusterUnavailable": {
      "Type": "Fail",
      "Cause": "Redshift cluster is not available",
      "Error": "Error"
    },
    "SalesETLPipelineComplete": {
      "Type": "Pass",
      "Next": "ValidateSalesMetric",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "validate_sales_metric",
          "sql_statement": [
            "select 1/count(1) from {0}.store_sales where ss_sold_date_sk in (select
d_date_sk from {0}.date_dim where d_date='{1}')"
          ]
        }
      }
    },
    "ValidateSalesMetric": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetValidateSalesMetricStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetValidateSalesMetricStatus": {
      "Type": "Task",
      "Next": "CheckValidateSalesMetricStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckValidateSalesMetricStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesMetricValidation"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "DataValidationComplete"
        }
      ],
      "Default": "SalesValidationWait"
    },
    "SalesValidationWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetValidateSalesMetricStatus"
    },
    "FailSalesMetricValidation": {
      "Type": "Fail",
      "Cause": "Data Validation Failed",
      "Error": "Error"
    },
```

```
    "DataValidationComplete": {
      "Type": "Pass",
      "Next": "InitializePauseCluster"
    },
    "InitializePauseCluster": {
      "Type": "Pass",
      "Next": "PauseCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "pause"
        }
      }
    },
    "PauseCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "PauseClusterWait",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus",
      "Catch": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "Next": "ClusterPausedComplete"
        }
      ]
    },
    "InitializeCheckPauseCluster": {
      "Type": "Pass",
      "Next": "GetStateOfPausedCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfPausedCluster": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterPaused",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterPaused": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializePauseCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "ClusterPausedComplete"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "PauseClusterWait"
        }
      ]
    },
    "PauseClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckPauseCluster"
    },
    "ClusterPausedComplete": {
      "Type": "Pass",
      "End": true
    }
  }
```

```
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:CFN36-RedshiftDataApi-
AIDACKCEVSQ6C2EXAMPLE",
                "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

使用Step Functions和AWS Batch錯誤處理

此範例專案示範如Step Functions何使用具有錯誤處理功能的狀態機器來執行AWS Batch作業。

在此專案中，Step Functions 會使用狀態機器同步呼叫 AWS Batch 任務。然後，它會等待工作成功或失敗，在工作失敗時重試並捕獲錯誤，然後傳送一個Amazon SNS主題，其中包含有關工作是成功還是失敗的消息。

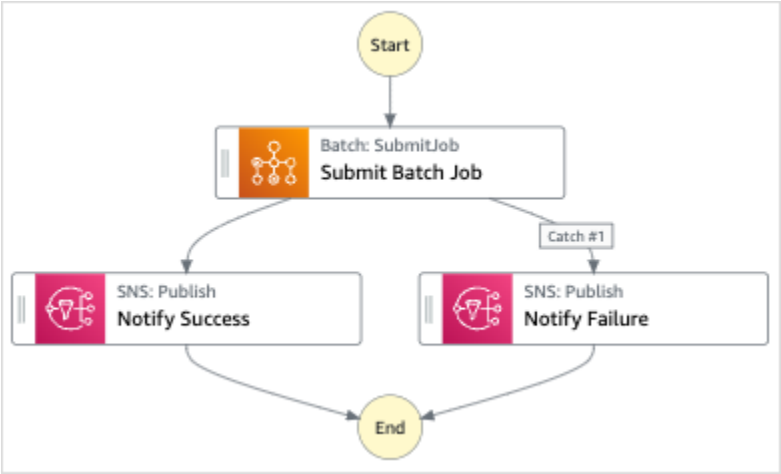
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Manage a batch job`**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇「管理批次工作」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 一 AWS Batch 份工作
   - Amazon SNS 主題
   - 一个 AWS Step Functions 状态机
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示「管理批次工作範例專案」的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀

態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 允許您為狀態機器，執行和活動以及包含非 ASCII 字符的標籤創建名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 AWS Batch 和 Amazon SNS 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN)，並傳遞至服務 API `Parameters` 來瞭解步驟函數如何控制 AWS Batch 和 Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch
 job completion",
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
```

```
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "arn:aws:batch:us-west-2:123456789012:job-queue/
BatchJobQueue-123456789abcdef",
        "JobDefinition": "arn:aws:batch:us-west-2:123456789012:job-definition/
BatchJobDefinition-123456789abcdef:1"
      },
      "Next": "Notify Success",
      "Retry": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "IntervalSeconds": 30,
          "MaxAttempts": 2,
          "BackoffRate": 1.5
        }
      ],
      "Catch": [
        {
          "ErrorEquals": [ "States.ALL" ],
          "Next": "Notify Failure"
        }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions succeeded",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions failed",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
```

```
      },
      "End": true
    }
  }
}
```

IAM 範例

範例專案產生的此範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器及相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

Example **BatchJobNotificationAccessPolicy**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
```

```
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

扇出一 AWS Batch 份工作

此範例專案示範如何使用 Step 函式的Map狀態來散播 AWS Batch 工作。

在此專案中，Step Functions 會使用狀態機器叫用 Lambda 函數來執行簡單的預處理，然後使用狀態 parallel 叫用多個 AWS Batch 作業。Map

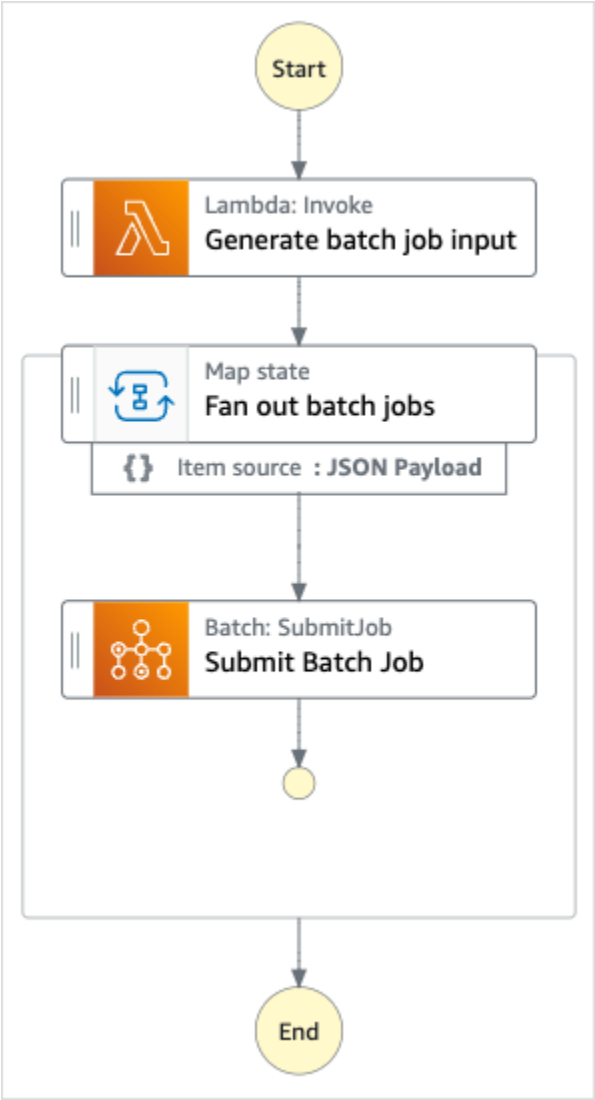
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **Fan out a batch job**在搜尋方塊中輸入，然後從傳回的搜尋結果中選擇展開批次工作。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - Lambda 函數
   - AWS Batch工作佇列
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下圖顯示「展開批次作業範例專案」的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

    1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

   > **Note**
   >
   > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 AWS Batch 和 Amazon SNS 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN)，並傳遞至服務 API `Parameters` 來瞭解步驟函數如何控制 AWS Batch 和 Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for fanning out AWS Batch job",
  "StartAt": "Generate batch job input",
  "TimeoutSeconds": 3600,
  "States": {
    "Generate batch job input": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<GENERATE_BATCH_JOB_INPUT_LAMBDA_FUNCTION_NAME>"
      },
      "Next": "Fan out batch jobs"
    },
    "Fan out batch jobs": {
      "Comment": "Start multiple executions of batch job depending on pre-processed
 data",
      "Type": "Map",
      "End": true,
      "ItemsPath": "$",
      "Parameters": {
        "BatchNumber.$": "$$.Map.Item.Value"
      },
      "Iterator": {
        "StartAt": "Submit Batch Job",
        "States": {
          "Submit Batch Job": {
            "Type": "Task",
            "Resource": "arn:aws:states:::batch:submitJob.sync",
            "Parameters": {
              "JobName": "BatchJobFanOut",
              "JobQueue": "<BATCH_QUEUE_ARN>",
              "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>"
            },
            "End": true
          }
        }
      }
    }
  }
}
```

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

Example **`BatchJobFanOutAccessPolicy`**

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Example **InvokeGenerateBatchJobMapLambdaPolicy**

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:StepFunctionsSample-BatchJobFa-
GenerateBatchJobMap-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

AWS Batch 與 Lambda

此範例專案示範如何使用 Step Functions 預先處理具有 AWS Lambda 函數的資料，然後協調 AWS Batch 工作。

在此專案中，Step Functions 會使用狀態機器叫用 Lambda 函數，在提交 AWS Batch 工作之前執行簡單的預處理。根據前一個結果或成功，可以調用多個作業。

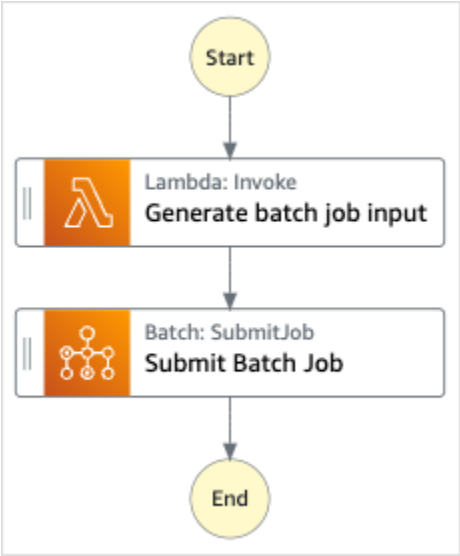
步驟 1：建立狀態機並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **`Batch job with Lambda`**在搜尋方塊中輸入，然後Lambda從傳回的搜尋結果中選擇「Batch處理工作」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - Lambda 函數
   - AWS Batch 任務
   - 一个 AWS Step Functions 状态机
   - 相關的 AWS Identity and Access Management (IAM) 角色

   下圖顯示具有Lambda範例專案之 Batch 工作的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到提供類似 VS Code 的整合式程式碼編輯器，以便在 Step Functions 主控台中更新狀態機器的 Amazon States Language (ASL) 定義。程式碼模式如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

     > **Important**
     >
     > 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

   - 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案會使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

     > **Tip**
     >
     > 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

     準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘的時間。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：

   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。默認情況下，Step Functions 自動生成一個唯一的執行名稱。

      > **Note**
      >
      > Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

   2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

      如果您選擇執行示範，則不需要提供任何執行輸入。

      > **Note**
      >
      > 如果您部署的示範專案包含預先填入的執行輸入資料，請使用該輸入來執行狀態機器。

   3. 選擇 Start execution (開始執行)。
   4. 「Step Functions」主控台會將您引導至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

範例狀態機器程式碼

此範例專案中的狀態機器會直接將參數傳遞至這些資源，與 AWS Batch 和 Amazon SNS 整合。

瀏覽此範例狀態機器，瞭解 Step Functions 如何透過連線至`Resource`欄位中的 Amazon 資源名稱 (ARN)，並傳遞至服務 API `Parameters` 來瞭解步驟函數如何控制 AWS Batch 和 Amazon SNS。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

```
{
  "Comment": "An example of the Amazon States Language for using batch job with pre-
processing lambda",
  "StartAt": "Generate batch job input",
  "TimeoutSeconds": 3600,
  "States": {
    "Generate batch job input": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.batch_input",
      "Parameters": {
        "FunctionName": "<GENERATE_BATCH_JOB_INPUT_LAMBDA_FUNCTION_NAME>"
      },
      "Next": "Submit Batch Job"
    },
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobFanOut",
        "JobQueue": "<BATCH_QUEUE_ARN>",
        "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>",
        "Parameters.$": "$.batch_input"
      },
      "End": true
    }
  }
```

```
}
```

IAM 範例

範例專案產生的這些範例 AWS Identity and Access Management (IAM) 政策包含執行狀態機器和相關資源所需的最低權限。我們建議您僅在 IAM 政策中加入必要的許可。

Example **BatchJobWithLambdaAccessPolicy**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:ManageBatchJob-SNSTopic-
JHLYYG7AZPZI"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
```

```
    ]
}
```

Example **InvokeGenerateBatchJobMapLambdaPolicy**

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:StepFunctionsSample-BatchWithL-
GenerateBatchJobMap-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

如需將 Step Functions 與其他 AWS 服務搭配使用時如何設定 IAM 的詳細資訊，請參閱整合式服務的 IAM 政策。

使用執行 AI 提示鏈結 Amazon Bedrock

此範例專案示範如何整合以執Amazon Bedrock行 AI 提示鏈結。此示例項目展示了如何使用Amazon Bedrock. 專案會將一些提示連結在一起，並按照提供的順序解析它們。這些提示的鏈結可增強用於提供高度策劃回應的語言模型的能力。

此範例專案會建立狀態機器、支援 AWS 資源，並設定相關的 IAM 許可。探索此範例專案，瞭解如何使用Amazon Bedrock最佳化服務整合與Step Functions狀態機器，或將其用作您自己專案的起點。

主題

- AWS CloudFormation 範本和其他資源
- 必要條件
- 步驟 1：建立狀態機器並佈建資源
- 步驟 2：運行狀態機

AWS CloudFormation 範本和其他資源

您可以使用CloudFormation範本來部署此範例專案。此範本會在您的中建立下列資源 AWS 帳戶：

- 狀Step Functions態機。
- 狀態機器的執行角色。此角色授 AWS 服務 與狀態機存取其他資源 (例如Amazon BedrockInvokeModel動作) 所需的權限。

必要條件

此示例項目使用 Cohere 命令大語言模型（LLM）。要成功運行此示例項目，您必須從Amazon Bedrock控制台添加對此 LLM 的訪問權限。欲新增模型存取權限，請執行下列操作：

1. 打開 Amazon 基岩控制台。
2. 在導覽窗格中，選擇 [模型存取權]。
3. 選擇管理模型存取權限。
4. 選取「共享」旁邊的核取方塊。
5. 選擇要求存取權限。Cohere 模型的訪問狀態顯示為授予訪問權限。

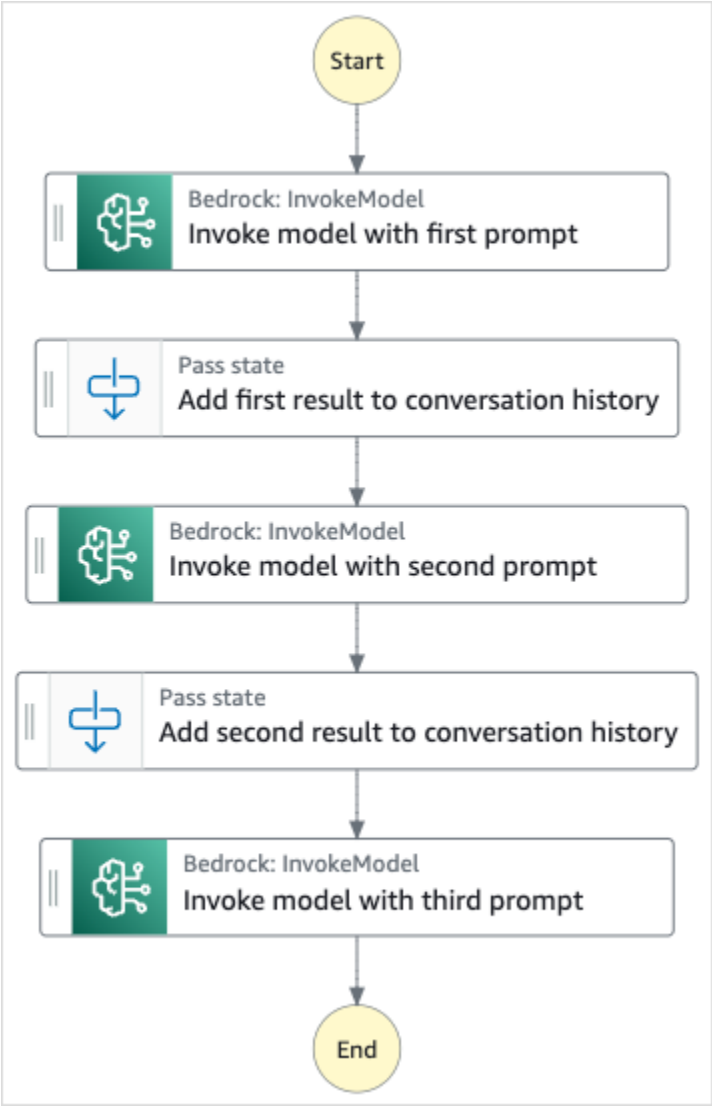
步驟 1：建立狀態機器並佈建資源

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. **bedrock**在搜尋方塊中輸入，然後Bedrock從傳回的搜尋結果中選擇「執行 AI 提示鏈結」。
3. 選擇 Next (下一步) 繼續。
4. Step Functions 會列出您選取的範例專案中 AWS 服務 使用的項目。它也會顯示範例專案的工作流程圖表。將此項目部署到您的項目中， AWS 帳戶 或將其用作構建自己的項目的起點。根據您想要的進行方式，選擇 [執行示範] 或 [在其上建置]。

   此範例專案會部署下列資源：

   - 一个 AWS Step Functions 状态机
   - 相關 AWS Identity and Access Management (IAM) 角色

   下圖顯示了使Bedrock用範例專案執行 AI 提示鏈結的工作流程圖形：



5. 選擇「使用範本」繼續進行選取。
6. 執行以下任意一項：

   - 如果您選取「在其上建置」，「Step Functions」會為您選取的範例專案建立工作流程原型。Step Functions 不會部署工作流程定義中列出的資源。

     在工作流程 Studio 中設計模式，從拖放狀態狀態瀏覽器以繼續建立您的工作流程原型。或者切換到程式碼模式提供類似 VS 代碼的集成代碼編輯器，用於在 Step Functions 控制台中更新狀態機器的Amazon States Language（ASL）定義。如需有關使用工作流程 Studio 建置狀態機器的詳細資訊，請參閱使用工作流程。

> **Important**
>
> 請記得在執行工作流程之前，更新範例專案中使用之資源的預留位置 Amazon 資源名稱 (ARN)。

- 如果您選取 [執行示範]，Step Functions 會建立唯讀範例專案，該專案使用 AWS CloudFormation 範本將該範本中列出的 AWS 資源部署到您的 AWS 帳戶.

> **Tip**
>
> 若要檢視範例專案的狀態機定義，請選擇 [程式碼]。

準備就緒後，請選擇 [部署並執行] 以部署範例專案並建立資源。

建立這些資源和相關 IAM 許可最多可能需要 10 分鐘。部署資源時，您可以開啟 CloudFormation Stack ID 連結以查看正在佈建的資源。

建立範例專案中的所有資源之後，您可以在 [狀態機器] 頁面上看到列出的新範例專案。

> **Important**
>
> CloudFormation 範本中使用的每項服務可能會收取標準費用。

步驟 2：運行狀態機

1. 在 [狀態電腦] 頁面上，選擇您的範例專案。
2. 在範例專案頁面上，選擇 [開始執行]。
3. 在 [開始執行] 對話方塊中，執行下列動作：
   1. (選擇性) 若要識別您的執行項目，您可以在「名稱」(Name) 方塊中指定執行項目的名稱。依預設，Step Functions 會自動產生唯一的執行名稱。

> **Note**
>
> Step Functions 可讓您為包含非 ASCII 字元的狀態機器、執行項目、活動和標籤建立名稱。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

2. (選擇性) 在「輸入」方塊中，以 JSON 格式輸入輸入值以執行工作流程。

   如果您選擇執行示範，則不需要提供任何執行輸入。

3. 選擇 Start execution (開始執行)。
4. Step Functions 主控台會將您導向至標題為執行 ID 的頁面。此頁面稱為「執行詳細資訊」頁面。在此頁面上，您可以在執行進行時或完成之後複查執行結果。

   若要複查執行結果，請在「圖形」檢視中選擇個別狀態，然後選擇步驟詳情窗格上的個別索引標籤，分別檢視每個狀態的詳細資訊，包括輸入、輸出和定義。如需有關可在「執行詳細資訊」頁面檢視之執行資訊的詳細資訊，請參閱執行詳細資訊頁面 — 介面概觀。

配額

AWS Step Functions 對某些狀態機參數的大小設置配額，例如特定時間段內的 API 動作數量或您可以定義的狀態機器數量。雖然這些配額的設計旨在預防設定錯誤的狀態機器使用系統的所有資源，但許多都不是硬性配額。

若要要求增加服務配額，您可以執行下列其中一項作業：

- 請使用 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/home。如需使用 Service Quotas 主控台要求增加配額的相關資訊，請參閱《Ser vice Quotas 使用者指南》中的要求增加配額。
- 使用中的 [Sup port 中心] 頁面，針對每個區域提供的資源要求增加配額。 AWS Management Console AWS Step Functions 如需詳細資訊，請參閱《AWS 》中的 AWS 一般參考服務配額。

> **Note**
>
> 如果狀態機器執行或活動執行的某個特定階段耗時過長，您可設定狀態機器逾時來觸發逾時事件。

主題

一般配額

配額	描述
Step Functions 中的名稱	狀態機器、執行項目和活動工作的名稱長度不得超過 80 個字元。這些名稱對於您的帳戶和 AWS 地區而言必須是唯一的，且不得包含以下任何一項： • 空白 • 萬用字元 (? *) • 括號字元 (< > { } [ ]) • 特殊字元 (: ; , \ | ^ ~ $ # % & ` ") • 控制字符 (\\u0000-\\u001f 或 \ \u007f-\\u009f) 。 如果您的狀態機器的類型為 Express，則可以為狀態機器的多個執行提供相同的名稱。即使多個執行具有相同的名稱，Step Functions 也會為每個 Express 狀態機器執行產生唯一的執行 ARN。 Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。

與帳戶相關的配額

資源	預設配額	可以提高
註冊狀態機器的數量上限	10,000	25,000
註冊活動的數量上限	10,000	15,000

資源	預設配額	可以提高
請求大小上限	每個請求 1 MB。這是每個 Step Functions 數 API 請求的總數據大小，包括請求標頭和所有其他關聯的請求數據。	硬配額
每個帳戶的開放執行上限	每人每次執行 100 萬次 AWS 帳戶 。 AWS 區域超過這個配額將導致 `Execution LimitExceeded` 錯誤。這不適用於快速工作流程。	百萬
開啟地圖執行次數上限 開放的地圖運行是一個已經開始但尚未完成的地圖運行。排定的地圖執行會在MapRunStarted事件中等待，直到開啟的地圖執行總數少於預設配額 1000 為止。	1000 此配額適用於「分散式地圖」狀態。	硬配額
地圖運行redrives的最大值。	1000 此配額適用於「分散式地圖」狀態。	硬配額

與 HTTP 工作相關的配額

HTTP 任務使用令牌存儲桶方案進行限制，以維護Step Functions服務帶寬。下表列出 HTTP 工作的儲存貯體大小和重新填寫率。

資源	儲存貯體大小	每秒重新填滿速率
HTTP 任務	300	300

下表列出 HTTP 工作持續時間的配額。

資源	預設配額
HTTP 任務持續時間 HTTP 任務持續時間是指 HTTP 任務發送 HTTP 請求和接收響應所花費的時間。	60 秒 這是無法變更的硬配額。

與狀態節流有關的配額

Step Functions 狀態轉換使用令牌存儲桶方案進行限制，以維護服務帶寬。標準工作流程和 Express 工作流程具有不同的狀態轉換節流。標準工作流程配額是軟配額，可以增加。

> **Note**
>
> `StateTransition`服務指標的節流報告為 Amazon `ExecutionThrottled` 中。CloudWatch如需詳細資訊，請參閱`ExecutionThrottled` CloudWatch 量度。

	Standard		Express	
服務指標	儲存貯體大小	每秒重新填滿速率	儲存貯體大小	每秒重新填滿速率
`StateTransition` — 美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 和歐洲 (愛爾蘭)	5,000	5,000	無限制	無限制
`StateTransition` — 所有其他地區	800	800	無限制	無限制

與 API 動作節流相關的配額

某些 Step Functions API 操作使用令牌存儲桶方案進行限制，以維護服務帶寬。這些配額是軟配額，可以增加。

> **Note**
>
> 節流配額是每個帳戶，每 AWS 個區域。 AWS Step Functions 隨時都可以增加鏟斗尺寸和補充率。

	Standard		Express	
API 名稱	儲存貯體大小	每秒重新填滿速率	儲存貯體大小	每秒重新填滿速率
`StartExecution` — 位於美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 和歐洲 (愛爾蘭)	1,300	300	6,000	6,000
`StartExecution` — 所有其他地區	800	150	6,000	6,000

與 TestState API 相關的配額

下表列出 TestState API 的可用配額。

API 名稱	配額	可以提高
TestState	一秒 1 個交易 (TPS)	硬配額

其他配額

這些配額是軟配額，可以增加。

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
API 名稱	儲存貯體大小	每秒重新填滿速率	儲存貯體大小	每秒重新填滿速率
CreateActivity	100	1	100	1
CreateStateMachine	100	1	100	1
DeleteActivity	100	1	100	1
DeleteStateMachine	100	1	100	1
DescribeActivity	200	1	200	1
DescribeExecution	300	15	250	10
DescribeStateMachine	200	20	200	20
DescribeStateMachineForExecution	200	1	200	1
GetActivityTask	3,000	500	1,500	300

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
API 名稱	儲存貯體大小	每秒重新填滿速率	儲存貯體大小	每秒重新填滿速率
GetExecutionHistory	400	20	400	20
ListActivities	100	10	100	5
ListExecutions	200	5	100	2
ListStateMachines	100	5	100	5
ListTagsForResource	100	1	100	1
SendTaskFailure	3,000	500	1,500	300
SendTaskHeartbeat	3,000	500	1,500	300
SendTaskSuccess	3,000	500	1,500	300

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
API 名稱	儲存貯體大小	每秒重新填滿速率	儲存貯體大小	每秒重新填滿速率
StartSync Execution	同步快速執行 API 呼叫不會造成現有帳戶容量限制。Step Functions 可依需求提供容量，並隨著持續的工作負載自動擴充工作負載激增可能會受到限制，直到容量可用為止。 如果您遇到節流狀態，請在一段時間後再試一次。如需同步快速工作流程的資訊，請參閱同步和非同步快速工作流。			
StopExecution	1,000	200	500	25
TagResource	200	1	200	1
UntagResource	200	1	200	1
UpdateStateMachine	100	1	100	1

與狀態機器執行相關的配額

下表說明與狀態機器執行相關的配額。狀態機器執行配額是無法變更的硬配額，除了執行歷程記錄保留時間配額。

配額	標準	Express
執行時間上限	1 年。如果執行的時間超過 1 年的最大值，則會失敗並顯示`States.Timeout` 錯誤並發出指標。`ExecutionsTimedOut` CloudWatch	5 分鐘。如果執行的執行時間超過 5 分鐘上限，則會失敗並顯示`States.Timeout` 錯誤並發出`ExecutionsTimedOut` CloudWatch 測量結果。
執行歷史記錄大小上限	單一狀態機器執行歷程記錄中的 25,000 個事件。如果執行歷史記錄達到此配額，執行將會失敗。若要避免這種情況，請參閱避免達到歷史記錄配額。	無限制。
執行閒置時間上限	1 年 (受最長執行時間限制)。	5 分鐘（受最大執行時間的限制）。
執行記錄保留時間	執行結束後 90 天。在這個時間之後，您就不能再擷取或檢視執行歷史記錄。步驟函數保留的已關閉執行次數沒有進一步的配額。 若要符合法規遵循、組織或法規需求，您可以傳送配額要求，將執行歷程記錄保留期限縮短為 30 天。若要這麼做，請使用 AWS Support Center Console 並建立新案例。 將保留期限縮短為 30 天的變更適用於區域中的每個帳戶。	若要查看執行歷史記錄 CloudWatch，必須設定 Amazon 日誌記錄。如需詳細資訊，請參閱 記錄使用CloudWatch日誌。

配額	標準	Express
執行redrivable期 Redrivable期間是指您可以redrive執行指定「標準工作流程」的時間。此期間從狀態機完成其執行之日開始。	十四天 此硬配額適用於「分散式地圖」狀態。	Redrive快速工作流程目前不支援。

與工作執行相關的配額

下表說明與工作執行相關的配額，並且是無法變更的硬配額。

配額	標準	Express
任務執行時間上限	1 年 (受限於最大執行時間)	5 分鐘 (受限於最大執行時間)
最長時間 Step Functions 將工作保留在佇列中	1 年 (受限於最大執行時間)	5 分鐘 (受限於最大執行時間)
每個 Amazon 資源名稱 (ARN) 的最大活動輪詢器	每個 ARN 1,000 個呼叫 `GetActivityTask` 的輪詢器。超過此配額會導致此錯誤：「已到達同時輪詢活動任務的工作者數量上限。」	不適用於快速工作流程。
工作、狀態或執行項目的最大輸入或輸出大小	256 KB 的資料作為一個 UTF-8 編碼的字串。此配額會影響排程工作、進入狀態或開始執行時的工作 (活動、Lambda 函數或整合式服務)、狀態或執行輸出，以及輸入資料。	256 KB 的資料作為一個 UTF-8 編碼的字串。此配額會影響排程工作、進入狀態或開始執行時的工作 (活動、Lambda 函數或整合式服務)、狀態或執行輸出，以及輸入資料。

與版本和別名相關的配額

資源	預設配額
已發佈的狀態機版本數目上限	每個狀態機 1000 個。 若要要求提高此軟限制，請使用中的「Sup port 中心」頁面AWS Management Console。
狀態機別名的最大數目	每個狀態機 100 個。 若要要求提高此軟限制，請使用中的「Sup port 中心」頁面AWS Management Console。

與標記相關的限制

標記 Step Functions 資源時，請注意這些限制。

> **Note**
>
> 標記限制無法像其他配額一樣增加。

限制	描述
每個資源的標籤數上限	50
金鑰長度上限	128 個 UTF-8 編碼的 Unicode 字元
數值長度上限	256 個 UTF-8 編碼的 Unicode 字元
字首限制	請勿在標籤名稱或值中使用`aws:`前置詞，因為它已保留供 AWS 使用。您不可編輯或刪除具此字首的標籤名稱或值。具此字首的標籤，不會算在每個資源配額的標籤計數內。
字元限制	標籤只能包含 Unicode 字母、數字、空格或這些符號：`_ . : / = + - @`

AWS Step Functions 中的記錄和監控

記錄和監控對於維護 Step Functions 和您的AWS解決方案的可靠性、可用性和效能非常重要。有幾種工具可與步驟函數搭配使用：

> **Tip**
>
> 若要將範例工作流程部署到您的，AWS 帳戶並瞭解如何監視工作流程執行的指標、記錄和追蹤，請參閱單元 12-AWS Step Functions工作坊的可觀察性。

主題

監視 Step Functions 使用 CloudWatch

監控是維持 AWS 解決方案的可靠性、可用性和效能的 AWS Step Functions 重要組成部分。您應該從使用的 AWS 服務收集盡可能多的監視資料，以便對多點失敗進行偵錯。開始監視 Step Functions 之前，您應該先建立可回答下列問題的監視計劃：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

下一個步驟是在您的環境中建立正常 效能的基準。若要完成此步驟，請在各種時間及不同負載條件下測量效能。監視 Step Functions 數時，請考慮儲存歷史監視資料。這類資料會提供基準，讓您可與目前的效能資料比較，以識別出正常的效能模式和效能異常狀況，再規劃方法來處理問題。

例如，使用 Step Functions 時，您可以監視有多少活動或 AWS Lambda 工作因活動訊號逾時而失敗。當效能超出建立的基準時，您可能需要變更活動訊號的間隔。

若要建立基準，您至少必須監控下列指標：

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

以下各節說明 Step Functions 提供給 Amazon 的指標 CloudWatch。您可以使用這些指標來追蹤狀態機器和活動，並設定閾值的警示。您可以使用檢視量度 AWS Management Console。

報告時間間隔的量度

某些「Step Functions」測 CloudWatch 量結果是時間間隔，一律以毫秒為單位。這些指標通常對應於您的執行階段，您可以使用描述性名稱來設定狀態機器、活動和 Lambda 函數逾時。

例如，`ActivityRunTime` 指標會測量活動開始執行後所需的完成時間。您可以為相同的期間設定逾時值。

在 CloudWatch 主控台中，如果您選擇平均值作為時間間隔測量結果的顯示統計資料，則可以取得最佳結果。

報告計數的量度

部分 Step Functions 數 CloudWatch 量度報表的結果為計數。例如，`ExecutionsFailed` 會記錄狀態機器執行的失敗次數。

Step Functions 為每個狀態機執行發出兩個`ExecutionsStarted`指標。這會導致`ExecutionsStarted`測量結果的SampleCount統計值顯示每次執行狀態機器時的值 2。SampleCount 統計資料會顯示以`ExecutionStarted=1`及執行完成的`ExecutionStarted=0`時間。

> **Tip**
>
> 我們建議您選取「總和」作為在 CloudWatch 主控台中報告計數的量度的顯示統計資料。

執行指標

AWS/States命名空間包含所有「Step Functions」執行的下列測量結果。這些是跨區域帳戶套用的無維度指標。

指標	描述
OpenExecutionCount	目前開啟的執行項目大約數目 — 您帳戶中目前正在進行的工作流程。 其目的是提供您工作流程何時接近最大執行限制的深入分析，以避免在呼叫StartExecution 或標準工作流程時RedriveExecution 發ExecutionLimitExceeded生錯誤。 量度取決於使用中的工作流程狀態轉換，因此在較低層級時，估計值可能與觀察到的執行中工作流程計數不一致。
OpenExecutionLimit	開啟的執行數目上限。如需詳細資訊，請參閱 與帳戶相關的配額。 此限制不適用於快速工作流程。

具有版本或別名的狀態機的執行指標

當您使用版本或別名執行狀態機器執行時，Step Functions 會發出以下指標。只有在節流執行的情況下才會發出ExecutionThrottled指標。這些指標將包括用StateMachineArn於識別特定狀態機器。

指標	描述
ExecutionTime	執行開始與關閉時間之間的間隔 (以毫秒為單位)。

指標	描述
ExecutionThrottled	已限制的StateEntered 事件和重試次數。這和 StateTransition 的調節有關。如需詳細資訊，請參閱 與狀態節流有關的配額。
ExecutionsAborted	中止或終止的執行數目。
ExecutionsFailed	失敗的執行次數。
ExecutionsStarted	已開始執行的數目。
ExecutionsSucceeded	成功完成的執行項目數目。
ExecutionsTimedOut	因任何原因逾時的執行次數。

快速工作流程的執行度量

AWS/States命名空間包含「Step Functions 快速工作流程」執行的下列度量。

指標	描述
ExpressExecutionMemory	快速工作流程使用的總記憶體。
ExpressExecutionBilledDuration	「快速工作流程」計費的持續時間。
ExpressExecutionBilledMemory	「快速工作流程」計費的使用記憶體量。

Redrive標準工作流程的執行指標

當你redrive一個狀態機執行，Step Functions 發出以下指標。

對於所有redriven執行，會發出Executions*指標。例如，假設redriven執行中止。此執行將為和發出非零數據點。RedrivenExecutionsAborted ExecutionsAborted

指標	描述
ExecutionsRedriven	redriven執行次數。
RedrivenExecutions Aborted	已取消或終止的redriven執行數目。
RedrivenExecutions TimedOut	因任何原因逾時的redriven執行次數。
RedrivenExecutions Succeeded	順利完成的redriven執行數目。
RedrivenExecutions Failed	失敗的redriven執行次數。

Step Functions 執行測量結果的維度

維度	描述
StateMachineArn	有問題執行的狀態機器的 Amazon 資源名稱 (ARN)。

具有版本的執行維度

維度	描述
StateMachineArn	執行由版本開始的狀態機器的 Amazon 資源名稱 (ARN)。
Version	用於啟動執行的狀態機器版本。

具有別名的執行維度

維度	描述
StateMachineArn	執行由別名開始的狀態機器的 Amazon 資源名稱 (ARN)。

維度	描述
Alias	用於啟動執行的狀態機器別名。

版本和別名的資源計數量

AWS/States命名空間包括狀態機器版本和別名計數的下列度量。

指標	描述
AliasCount	為狀態機器建立的別名數目。 每個狀態機最多可以建立 100 個別名。
VersionCount	針對狀態機器發行的版本數目。 您最多可以發佈 1000 個版本的狀態機器。

版本和別名之資源計數量度的維度

維度	描述
ResourceArn	具有版本或別名的狀態機器的 Amazon 資源名稱（ARN）。

活動指標

AWS/States命名空間包含「Step Functions」活動的下列測量結果。

指標	描述
ActivityRunTime	活動開始與關閉時間之間的間隔 (以毫秒為單位)。
ActivityScheduleTime	活動保持在排程狀態的間隔 (以毫秒為單位)。
ActivityTime	從活動排程到活動關閉的時間之間的間隔，以毫秒為單位。

指標	描述
ActivitiesFailed	失敗的活動數目。
ActivitiesHeartbeatTimedOut	因活動訊號逾時而逾時的活動數目。
ActivitiesScheduled	已排程活動的數目。
ActivitiesStarted	已開始活動的數目。
ActivitiesSucceeded	成功完成的活動數目。
ActivitiesTimedOut	關閉時逾時的活動數目。

Step Functions 活動測量結果的維度

維度	描述
ActivityArn	該活動的 ARN。

Lambda 功能指標

命AWS/States名空間包括步驟函數 Lambda 函數的下列量度。

指標	描述
LambdaFunctionRunTime	Lambda 函數啟動到關閉時間之間的間隔 (以毫秒為單位)。
LambdaFunctionScheduleTime	Lambda 函數保持在排程狀態的間隔 (以毫秒為單位)。
LambdaFunctionTime	Lambda 函數排定到關閉時間之間的間隔 (以毫秒為單位)。
LambdaFunctionsFailed	失敗的 Lambda 函數數目。

指標	描述
LambdaFunctionsScheduled	排程的 Lambda 函數數。
LambdaFunctionsStarted	已啟動的 Lambda 函數數目。
LambdaFunctionsSucceeded	成功完成的 Lambda 函數數目。
LambdaFunctionsTimedOut	關閉時逾時的 Lambda 函數數目。

步驟函數 Lambda 函數量度量的維度

維度	描述
LambdaFunctionArn	Lambda 函數的 ARN。

> **Note**
>
> 系統會針對在欄位中指定 Lambda 函數 ARN 的工作狀態發出 Lambda 函數指標。`Resource`改為使用`"Resource": "arn:aws:states:::lambda:invoke"`發出服務整合度量的工作狀態。如需詳細資訊，請參閱 使用 Step Functions 叫用 Lambda。

服務整合指標

命`AWS/States`名空間包含下列 Step Functions 服務整合的量度。如需詳細資訊，請參閱 AWS Step Functions 搭配其他服務使用。

指標	描述
ServiceIntegrationRunTime	服務工作開始與關閉時間之間的間隔 (以毫秒為單位)。

指標	描述
ServiceIntegrationScheduleTime	服務工作保持在排程狀態的間隔 (以毫秒為單位)。
ServiceIntegrationTime	從排定服務工作到關閉時間之間的間隔 (以毫秒為單位)。
ServiceIntegrationsFailed	失敗的服務作業數目。
ServiceIntegrationsScheduled	排程的服務作業數目。
ServiceIntegrationsStarted	已啟動的服務作業數目。
ServiceIntegrationsSucceeded	順利完成的服務作業數目。
ServiceIntegrationsTimedOut	關閉時逾時的服務作業數目。

Step Functions 服務整合測量結果的維度

維度	描述
ServiceIntegrationResourceArn	整合服務的資源 ARN。

服務指標

AWS/States命名空間包含「Step Functions」服務的下列測量結果。

指標	描述
ThrottledEvents	已限制的要求計數。

指標	描述
ProvisionedBucketSize	每秒可用要求的計數。
ProvisionedRefillRate	允許進入值區的每秒要求計數。
ConsumedCapacity	每秒要求計數。

Step Functions 服務測量結果的維度

維度	描述
ServiceName	篩選資料以顯示狀態轉換指標

API 指標

命AWS/States名空間包含 Step Functions API 的下列量度。

指標	描述
ThrottledEvents	已限制的要求計數。
ProvisionedBucketSize	每秒可用要求的計數。
ProvisionedRefillRate	允許進入值區的每秒要求計數。
ConsumedCapacity	每秒要求計數。

Step Functions API 指標的維度

維度	描述
APIName	將資料篩選為指定之 API 名稱的 API。

最大努力 CloudWatch 指標交付

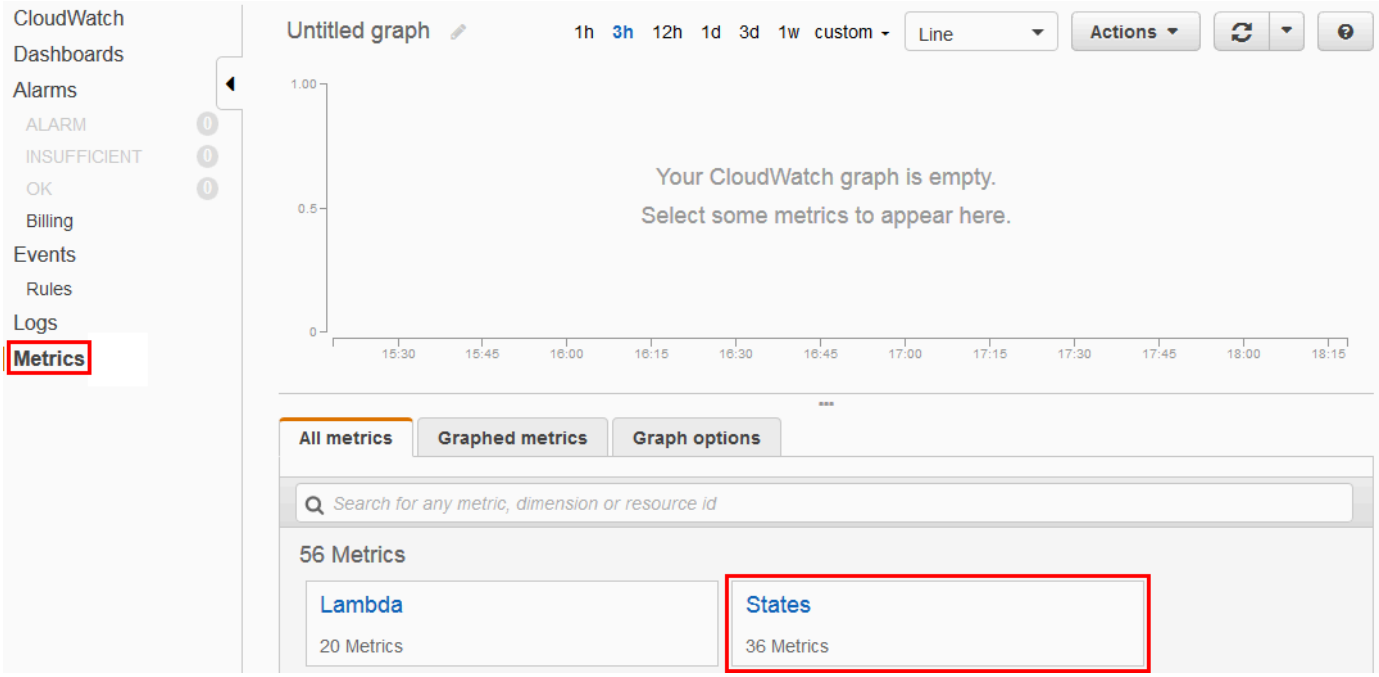
CloudWatch 指標是依最佳作業基礎交付。

不保證指標的完成程度與時間先後順序。特定要求的資料點回傳，回傳時附有的時間戳記可能會晚於實際處理要求時間。資料點可能會延遲一分鐘，然後才能透過使用 CloudWatch，否則可能根本無法傳遞。 CloudWatch請求指標可讓您以近乎即時的方式瞭解狀態機器執行情況。這並不意味著是所有與執行相關的指標的完整會計。

根據此功能的最大努力，帳單和成本管理儀表板上可用的報表可能包含一或多個未出現在執行指標中的存取要求。

檢視 Step Functions 的測量結果

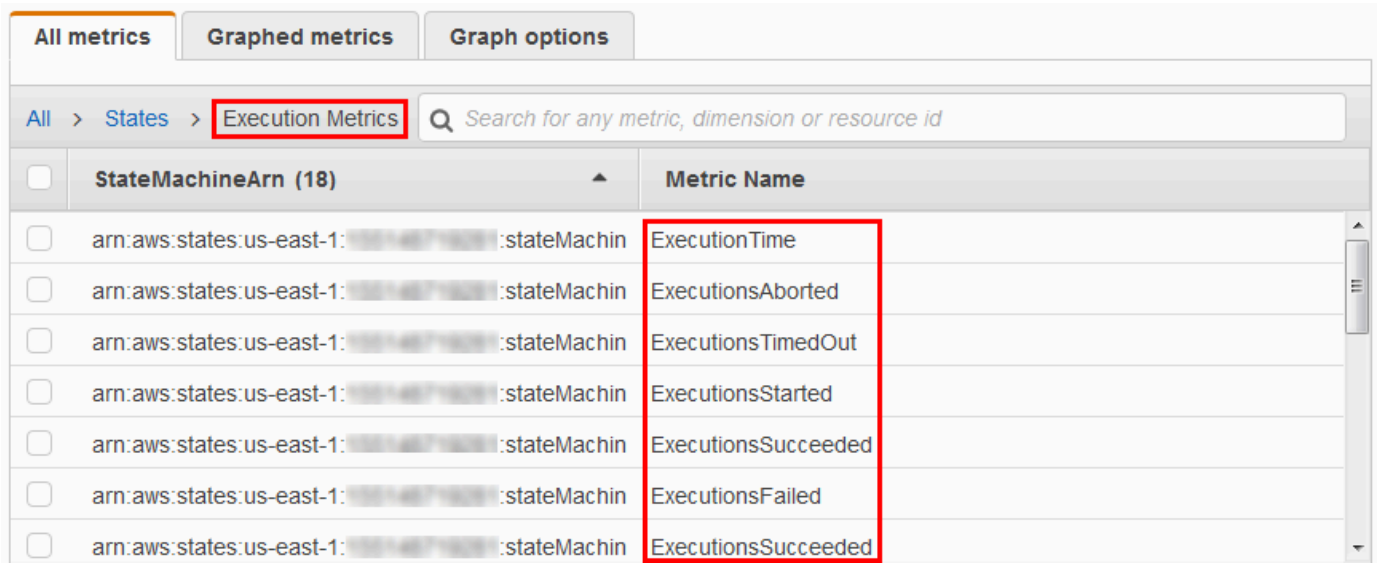
1. 登入 AWS Management Console 並開啟 CloudWatch 主控台。
2. 選擇 Metrics (指標) 並前往 All Metrics (所有指標) 標籤，然後選擇 States (狀態)。



如果您最近執行了任何執行，最多可以看到四種類型的指標：

- 執行指標
- Activity Function Metrics (活動函數指標)
- Lambda 函數指標
- 服務整合指標

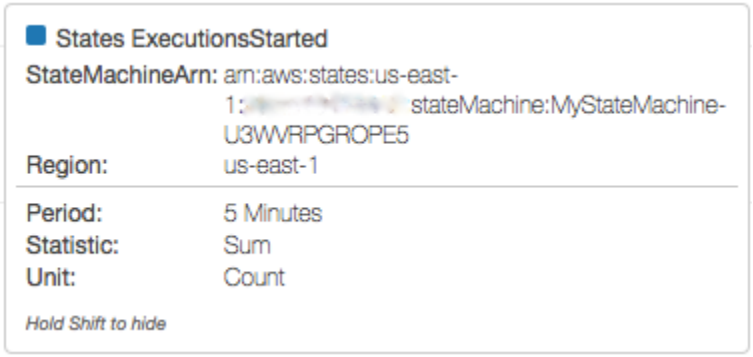
3. 選擇指標類型，以查看指標清單。



- 若要依量度名稱排序量度 StateMachineArn，或使用欄標題。
- 若要檢視指標的圖表，請在清單上選擇指標旁的方塊。您可以使用圖表視圖上方的時間範圍控制變更圖表參數。

  您可以使用相對值或絕對值 (特定天數和時間) 選擇自訂時間範圍。您還可以使用下拉式清單將值顯示為行、重疊圖表或號碼 (值)。
- 若要查看圖表的詳細資訊，請將滑鼠游標移到圖表下方顯示的指標顏色代碼上。

  該指標的詳細資訊隨即顯示。



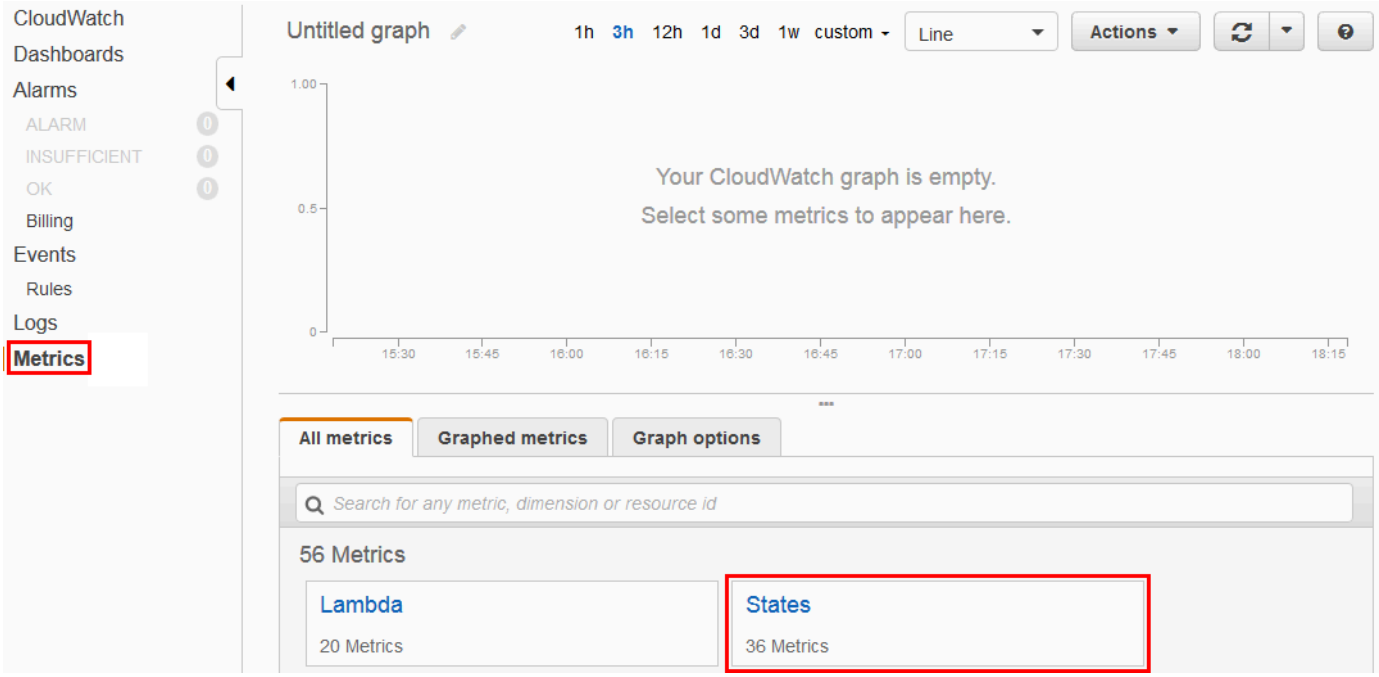
如需使用指 CloudWatch 標的詳細資訊，請參閱 Amazon 使用者指南中的使 CloudWatch 用 Amazon 指 CloudWatch 標。

設定 Step Functions 的警報

您可以使用 Amazon CloudWatch 警示來執行動作。例如，如果您想知道何時達到警示臨界值，可以設定警示，以便在`StateMachinesFailed`指標上升超過特定閾值時傳送通知給 Amazon SNS 主題，或傳送電子郵件。

設定指標的警示

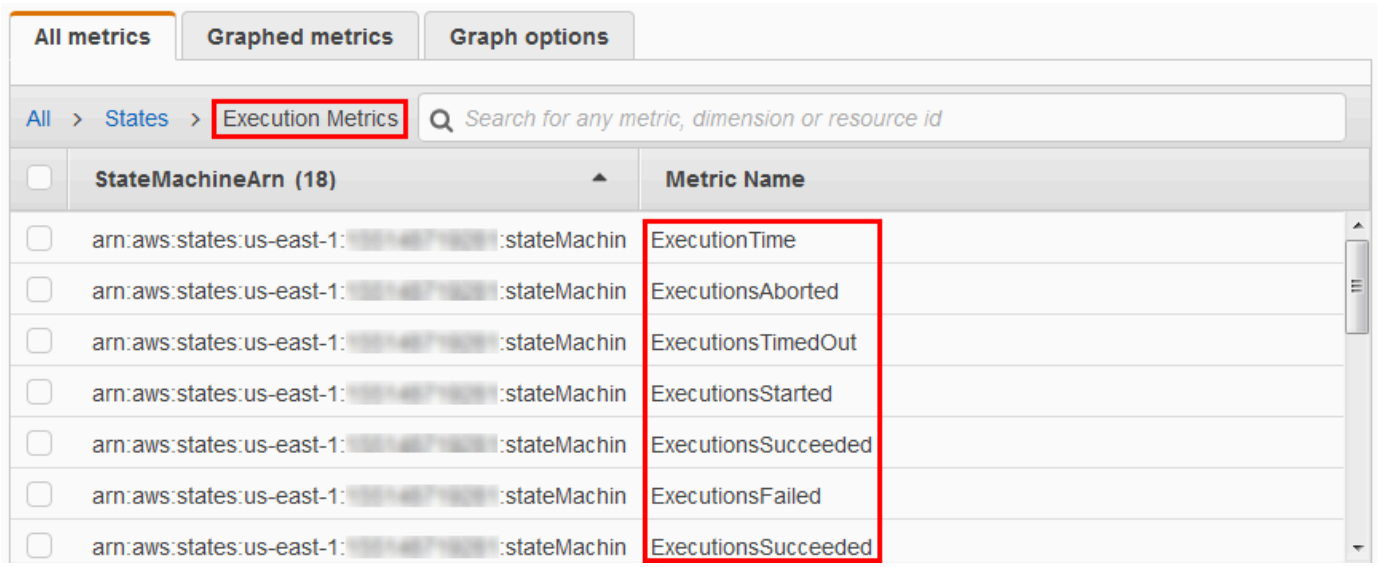
1. 登入 AWS Management Console 並開啟 CloudWatch 主控台。
2. 選擇 Metrics (指標) 並前往 All Metrics (所有指標) 標籤，然後選擇 States (狀態)。



如果您最近執行了任何執行，最多可以看到四種類型的指標：

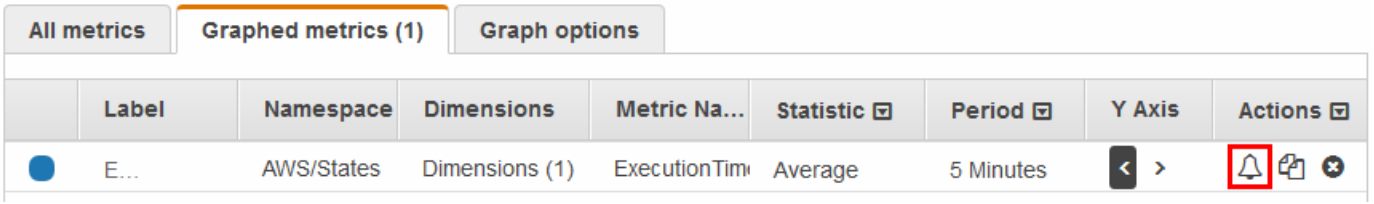
- 執行指標
- Activity Function Metrics (活動函數指標)
- Lambda 函數指標
- 服務整合指標

3. 選擇指標類型，以查看指標清單。

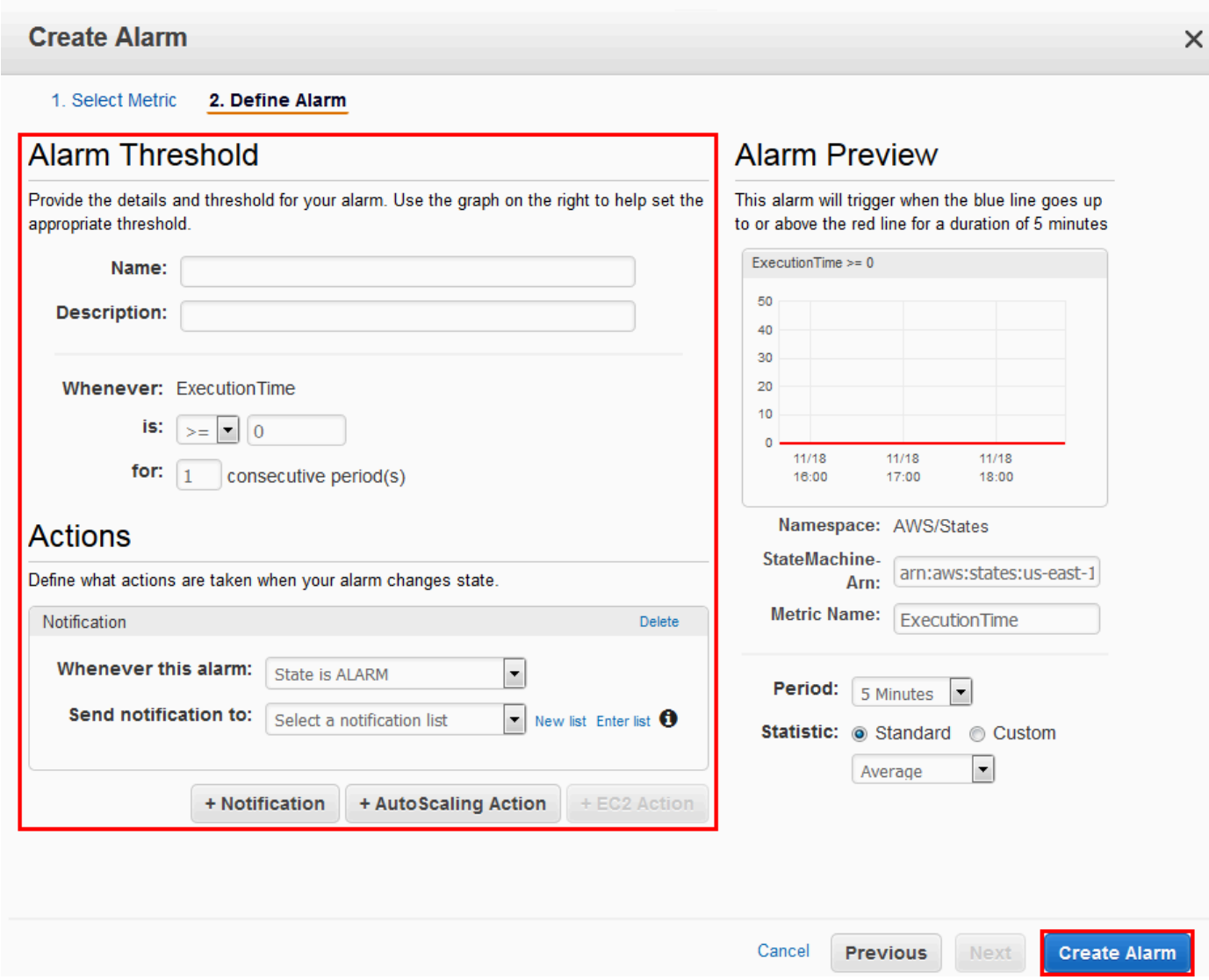


4. 選擇指標，然後選擇 Graphed metrics (圖表化指標)。
5. 在清單中選擇指標旁邊的

。



即會顯示 Create Alarm (建立警示) 頁面。



6. 輸入 Alarm threshold (警示閾值) 和 Actions (動作) 的值，然後選擇 Create Alarm (建立警示)。

如需有關設定和使用 CloudWatch 警示的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的建立 Amazon CloudWatch 警示。

EventBridge (CloudWatch 事件) 用於 Step Functions 執行狀態變更

Amazon EventBridge 是一項 AWS 服務，可讓您回應資 AWS 源中的狀態變更。例如，您可以 EventBridge 使用下列兩種方式來回應「Step Functions 標準工作流程」的執行狀態變更：

- 您可以設定 EventBridge 規則來回應 Step Functions 狀態機器的執行狀態變更時所發出的事件。這可讓您監控工作流程，而不需使用 DescribeExecution API 來持續輪詢。根據狀態機器執行的變

更，您可以使用 EventBridge 目標來啟動新的狀態機執行、呼叫 AWS Lambda 函數、將訊息發佈到 Amazon Simple Notification Service (Amazon SNS) 主題等。

- 您也可以將「Step Functions」狀態機器設定為中的目標 EventBridge。這可讓您觸發「Step Functions」工作流程的執行，以回應來自其他 AWS 服務的事件。

  如需詳細資訊，請參閱 Amazon EventBridge 使用者指南。

但是，Express 工作流程不會向其發出事件 EventBridge。若要監視「快速工作流程」的執行，您可以使用「 CloudWatch 記錄檔」。若要這樣做，請在狀態機器執行詳細資訊頁面上，選擇監視和記錄標籤。在監視索引標籤上，您可以檢視事件的 CloudWatch 指標，例如執行持續時間、執行錯誤和計費記憶體。在 [記錄] 索引標籤上，您可以檢視最近的記錄和記錄設定。

> **Tip**
>
> 若要將 Express 工作流程的範例部署到您的， AWS 帳戶 並瞭解如何監視 Express 工作流程，請參閱 AWS Step Functions 工作坊的監視 Express 工作流程模組。

EventBridge 有效載荷

EventBridge 事件可以在其定義中包含 input 屬性。對於某些事件， EventBridge事件也可以在其定義中包含輸出屬性。

- 如果發送的組合轉義輸入和轉義輸出 EventBridge 超過 248KB，則輸入將被排除。同樣地，如果逸出輸出超過 248KB，則會排除輸出。這是 EventBridge事件配額的結果。
- 您可以判斷裝載是否已使用`inputDetails`和`outputDetails`屬性截斷。如需詳細資訊，請參閱資`CloudWatchEventsExecutionDataDetails`料類型。
- 對於標準工作流程，您可以使用查看完整的輸入和輸出DescribeExecution。
- `DescribeExecution`不適用於快速工作流程。如果您想查看完整的輸入/輸出，可以使用「標準工作流程」來包裝「快速工作流程」。另一種選擇是使用 Amazon S3 ARN。如需有關使用 ARN 的資訊，請參閱the section called “使用 Amazon S3 ARN 而不是傳遞大型有效載荷”。

主題

- Step Functions 事件範例
- 將 Step Functions 事件路由至 EventBridge 主 EventBridge 控台

Step Functions 事件範例

以下是將事件傳送至的 Step Functions 的範例 EventBridge：

主題

- 已開始執行
- 執行成功
- 執行失敗
- 執行逾時
- 已中止執行

在每個案例中，在事件資料中的 `detail` 區段會提供與 `DescribeExecution` API 相同的資訊。`status` 欄位指出事件傳送時的執行狀態，根據發出的事件，狀態為以下其中之一：`RUNNING`、`SUCCEEDED`、`FAILED`、`TIMED_OUT` 或 `ABORTED`。

已開始執行

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws::states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "RUNNING",
        "startDate": 1551225271984,
        "stopDate":  null,
        "input": "{}",
```

```
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

執行成功

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "SUCCEEDED",
        "startDate": 1547148840101,
        "stopDate": 1547148840122,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": "\"Hello World!\"",
        "outputDetails": {
             "included": true
        }
    }
}
```

執行失敗

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "FAILED",
        "startDate": 1551225146847,
        "stopDate": 1551225151881,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

執行逾時

```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
```

```
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "TIMED_OUT",
        "startDate": 1551224926156,
        "stopDate": 1551224927157,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

已中止執行

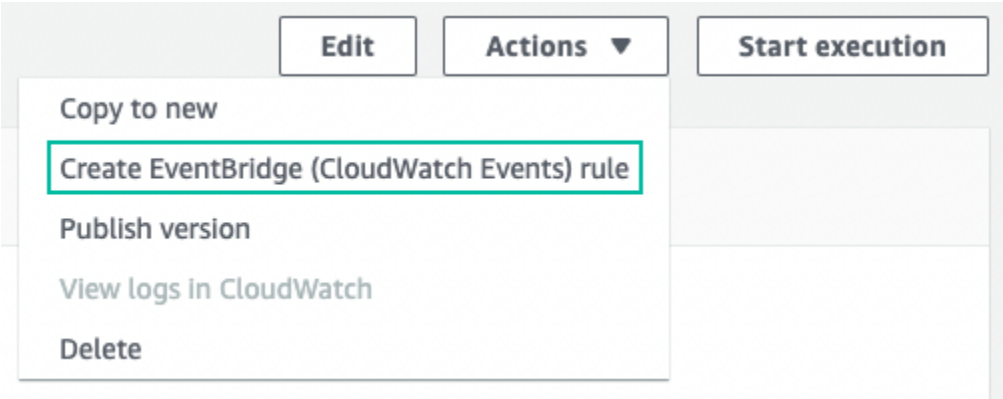
```
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "123456789012",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-2",
    "resources": [
      "arn:aws:states:us-east-2:123456789012:execution:state-machine-name:execution-
name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-2:123456789012:execution:state-machine-
name:execution-name",
        "stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:state-
machine",
        "name": "execution-name",
        "status": "ABORTED",
        "startDate": 1551225014968,
```

```
        "stopDate": 1551225017576,
        "input": "{}",
        "inputDetails": {
             "included": true
        },
        "output": null,
        "outputDetails": null
    }
}
```

將 Step Functions 事件路由至 EventBridge 主 EventBridge 控台

使用下列指示，瞭解如何在特定 Step Functions 狀態機器成功完成執行時觸發 Step Functions 狀態機器的執行。您可以使用 Amazon EventBridge 主控台指定要觸發其執行的狀態機器。

1. 在狀態機器的 [詳細資料] 頁面上，選擇 [動作]，然後選擇 [建立 EventBridge (CloudWatch 事件) 規則]。



   或者，您也可以在 https://console.aws.amazon.com/events/ 開啟 EventBridge 主控台。在導覽窗格中，選擇「匯流排」下的「規則」。
2. 選擇 Create rule (建立規則)。這會開啟 [定義規則詳細資訊] 頁面。
3. 輸入規則的「名稱」(例如，*StepFunctionsEventRule*)，並選擇性地輸入規則的「摘要」。
4. 對於事件匯流排和規則類型，請保留預設選項。
5. 選擇下一步。這會開啟 [建置事件模式] 頁面。
6. 在「事件來源」下，保留事件或 EventBridge 合作夥伴事件的AWS 預設選項。
7. 保留「範例」事件和「建立方法」區段的預設選取項。
8. 在事件模式下，執行下列操作：

   a. 在 [事件來源] 下拉式清單中，保留預設的AWS 服務選項。

b. 從AWS 服務下拉式清單中，選擇 Step Functions。

c. 從 [事件類型] 下拉式清單中，選取 [Step Functions 執行狀態變更]。

d. (選擇性) 設定特定狀態、狀態機器 Amazon 資源名稱 (ARN) 或執行 ARN。對於此程序，請選擇 [特定狀態]，然後從下拉式清單中選擇 [成功]。

9. 選擇下一步。這會開啟 [選取目標] 頁面。
10. 在 [目標類型] 下，保留預設的AWS 服務選項。
11. 從「選取目標」下拉式清單中，選擇 AWS 服務。例如，您可以啟動 Lambda 函數，或執行 Step Functions 數狀態機器。對於此程序，請選擇 Step Functions 狀態機。
12. 從狀態機下拉式清單中，選擇狀態機器。
13. 在 [執行角色] 下，保留 [為此特定資源建立新角色] 的預設選項。
14. 選擇下一步。這會開啟 [設定標籤] 頁面。
15. 再次選擇下一步。這會開啟 [檢閱並建立] 頁面。
16. 檢閱規則的詳細資訊，然後選擇 Create rule (建立規則)。

    規則隨即建立並顯示「規則」頁面，列出您的所有 Amazon EventBridge 規則。

記錄步驟函數使用 AWS CloudTrail

步驟函數集成了AWS CloudTrail，提供由用戶，角色或步驟功能中的服務採取的操作記錄的AWS服務。CloudTrail擷取步驟函式的所有 API 呼叫做為事件，包括來自 Step Functions 主控台的呼叫，以及從步驟函式 API 的程式碼呼叫。

如果您建立追蹤，您可以啟用連續交付CloudTrail事件到 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體，包括步驟函數的事件。如果您不設定追蹤記錄，仍然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新的事件。

使用收集的資訊CloudTrail，您可以判斷對 Step Functions 提出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱 AWS CloudTrail 使用者指南。

步驟函數資訊 CloudTrail

當您建立帳戶時，系統會在您的 AWS 帳戶中啟用 CloudTrail。當活動發生在步驟函數中時，該活動會與事件歷史記錄中的其他AWS服務CloudTrail事件一起記錄在事件中。

您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

如需AWS帳戶中持續記錄事件 (包括 Step Functions 的事件)，請建立追蹤。追蹤可CloudTrail將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析並根據 CloudTrail 日誌中所收集的事件資料採取行動。

如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 接收多個區域的 CloudTrail 日誌檔案及接收多個帳戶的 CloudTrail 日誌檔案

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：

- 該請求是否使用根或 使用者登入資料提出
- 提出該請求時，是否使用了特定角色或聯合身分使用者的臨時安全憑證
- 該請求是否由另一項 AWS 服務提出

如需詳細資訊，請參閱 CloudTrail 使用者身分元素。

範例：步驟函數記錄檔項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一個或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

CreateActivity

以下範例顯示的是展示 `CreateActivity` 動作的 CloudTrail 日誌項目。

```
{
    "eventVersion": "1.04",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:17:56Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "CreateActivity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "name":
 "OtherActivityPrefix.2016-10-27-18-16-56.894c791e-2ced-4cf4-8523-376469410c25"
    },
    "responseElements": {
        "activityArn": "arn:aws:states:us-
east-1:123456789012:activity:OtherActivityPrefix.2016-10-27-18-16-56.894c791e-2ced-4cf4-8523-37
        "creationDate": "Oct 28, 2016 1:17:56 AM"
    },
    "requestID": "37c67602-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "dc3becef-d06d-49bf-bc93-9b76b5f00774",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

CreateStateMachine

以下範例顯示的是展示 `CreateStateMachine` 動作的 CloudTrail 日誌項目。

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
        "userName": "test-user"
    },
```

```
    "eventTime": "2016-10-28T01:18:07Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "CreateStateMachine",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "name": "testUser.2016-10-27-18-17-06.bd144e18-0437-476e-9bb",
        "roleArn": "arn:aws:iam::123456789012:role/graphene/tests/graphene-execution-
role",
        "definition": "{   \"StartAt\": \"SinglePass\",   \"States\": {
  \"SinglePass\": {           \"Type\": \"Pass\",           \"End\": true       }   }}"
    },
    "responseElements": {
        "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:testUser.2016-10-27-18-17-06.bd144e18-0437-476e-9bb",
        "creationDate": "Oct 28, 2016 1:18:07 AM"
    },
    "requestID": "3da6370c-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "84a0441d-fa06-4691-a60a-aab9e46d689c",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

DeleteActivity

以下範例顯示的是展示 `DeleteActivity` 動作的 CloudTrail 日誌項目。

```
{
      "eventVersion": "1.04",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "AIDAJYDLDBVBI4EXAMPLE",
          "arn": "arn:aws:iam::123456789012:user/test-user",
          "accountId": "123456789012",
          "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
          "userName": "test-user"
      },
      "eventTime": "2016-10-28T01:18:27Z",
      "eventSource": "states.amazonaws.com",
      "eventName": "DeleteActivity",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "AWS Internal",
```

```
      "userAgent": "AWS Internal",
      "requestParameters": {
          "activityArn": "arn:aws:states:us-
east-1:123456789012:activity:testUser.2016-10-27-18-11-35.f017c391-9363-481a-be2e-"
      },
      "responseElements": null,
      "requestID": "490374ea-9cac-11e6-aed5-5b57d226e9ef",
      "eventID": "e5eb9a3d-13bc-4fa1-9531-232d1914d263",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
  }
```

DeleteStateMachine

以下範例顯示的是展示 DeleteStateMachine 動作的 CloudTrail 日誌項目。

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJABK5MNKNAEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/graphene/tests/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJA2ELRVCPEXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:17:37Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "DeleteStateMachine",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "errorCode": "AccessDenied",
    "errorMessage": "User: arn:aws:iam::123456789012:user/graphene/tests/test-user is
 not authorized to perform: states:DeleteStateMachine on resource: arn:aws:states:us-
east-1:123456789012:stateMachine:testUser.2016-10-27-18-16-38.ec6e261f-1323-4555-9fa",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "2cf23f3c-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "4a622d5c-e9cf-4051-90f2-4cdb69792cd8",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

StartExecution

以下範例顯示的是展示 StartExecution 動作的 CloudTrail 日誌項目。


```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:17:25Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "StartExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "input": "{}",
        "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:testUser.2016-10-27-18-16-26.482bea32-560f-4a36-bd",
        "name": "testUser.2016-10-27-18-16-26.6e229586-3698-4ce5-8d"
    },
    "responseElements": {
        "startDate": "Oct 28, 2016 1:17:25 AM",
        "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:testUser.2016-10-27-18-16-26.482bea32-560f-4a36-
bd:testUser.2016-10-27-18-16-26.6e229586-3698-4ce5-8d"
    },
    "requestID": "264c6f08-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "30a20c8e-a3a1-4b07-9139-cd9cd73b5eb8",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


StopExecution

以下範例顯示的是展示 StopExecution 動作的 CloudTrail 日誌項目。


```
{
```

```
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAJL5C75K4ZEXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2016-10-28T01:18:20Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "StopExecution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:testUser.2016-10-27-18-17-00.337b3344-83:testUser.2016-10-27-18-1
    },
    "responseElements": {
        "stopDate": "Oct 28, 2016 1:18:20 AM"
    },
    "requestID": "4567625b-9cac-11e6-aed5-5b57d226e9ef",
    "eventID": "e658c743-c537-459a-aea7-dafb83c18c53",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


記錄使用CloudWatch日誌

標準工作流程會記錄執行歷程AWS Step Functions，雖然您可以選擇將日誌記錄配置到亞馬遜CloudWatch記錄檔。

快速工作流程與標準工作流程不同，它不會在 AWS Step Functions 中記錄執行歷史記錄。若要查看快速工作流程的執行歷史記錄和結果，您必須設定 Amazon 的記錄CloudWatch記錄檔。發佈日誌不會阻擋執行或降低執行速度。

> **Note**
>
> 當您設定記錄時，CloudWatch記錄費用將適用，並按付費日誌費率向您收費。如需詳細資訊，請參閱付費記錄下日誌」頁籤上的CloudWatch定價頁面。

設定 記錄

當您使用 Step Functions 主控台建立標準工作流程時，它不會設定為啟用記錄CloudWatch記錄檔。依預設，會將使用步驟函數主控台建立的 Express 工作流程設定為啟用記錄CloudWatch記錄檔。

對於 Express 工作流程，步驟函數可以建立具有必要項目的角色AWS Identity and Access Management下列項目的 (IAM) 政策CloudWatch記錄檔。如果您使用 API、CLI 或建立「標準工作流程」或「快速工作流程」AWS CloudFormation，Step Functions 預設不會啟用記錄功能，而且您需要確保您的角色具有必要的權限。

對於從主控台啟動的每個執行，Step Functions 都會提供一個連結CloudWatch記錄檔，使用正確的篩選器設定，以擷取特定於該執行的記錄事件。

若要設定記錄，您可以傳遞LoggingConfiguration使用時的參數CreateStateMachine或者UpdateStateMachine。您可以進一步分析您的資料CloudWatch使用記錄CloudWatch日誌見解。如需詳細資訊，請參分析記錄資料CloudWatch日誌洞察。

CloudWatch記錄有效載荷

執行歷程記錄事件的定義中可能包含輸入或輸出屬性。如果轉義輸入或轉義輸出發送到CloudWatch記錄檔超過 248KB，因此會遭截斷CloudWatch記錄配額。

- 您可以檢閱以判斷裝載是否已被截斷`inputDetails`和`outputDetails`屬性。如需詳細資訊，請參閱`HistoryEventExecutionDataDetails`資料類型。
- 對於標準工作流程，您可以使用`GetExecutionHistory`。
- `GetExecutionHistory`不適用於快速工作流程。如果您想要查看完整的輸入和輸出，可以使用 Amazon S3 ARN。如需詳細資訊，請參閱the section called “使用 Amazon S3 ARN 而不是傳遞大型有效載荷”。

用於登入的 IAM 政策CloudWatch日誌

您還需要配置狀態機的執行 IAM 角色，以獲得適當的登錄權限CloudWatch記錄檔，如下列範例所示。

IAM 政策範例

下列是您可用來設定許可的範例政策。如下面的例子所示，你需要指定*在`Resource`字段，因為CloudWatchAPI 動作，例如CreateLogDelivery和DescribeLogGroups，不支持定義的資源類型 Amazon CloudWatch Logs。如需詳細資訊，請參閱由定義的動作Amazon CloudWatch Logs。

- 如需相關資訊CloudWatch資源，請參閱CloudWatch Logs資源與營運在亞馬遜CloudWatch使用者指南。
- 如需設定傳送記錄檔所需權限的相關資訊CloudWatch記錄檔，請參閱使用者權限在標題為的部分記錄檔傳送至CloudWatch Logs。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:CreateLogStream",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutLogEvents",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

無法訪問CloudWatch日誌

如果您無法存取CloudWatch日誌，請確保您已完成以下操作：

1. 將狀態機的執行 IAM 角色配置為具有登錄的適當權限CloudWatch記錄檔。

   如果您正在使用CreateStateMachine或者UpdateStateMachine請求，請確定您已在`roleArn`包含權限的參數，如前面的例子。

2. 檢查了CloudWatch記錄檔資源策略不超過 5120 個字元限制CloudWatch記錄資源策略。

   如果您已超過字元限制，請移除不必要的權限CloudWatch記錄資源策略，或在記錄群組名稱前面加上`/aws/vendedlogs`，這會將權限授與記錄群組，而不會在資源策略中附加更多字元。當您

在 Step Functions 主控台中建立記錄群組時，記錄群組名稱的前面會加上/aws/vendedlogs/states。如需詳細資訊，請參閱亞馬遜 CloudWatch 日誌資源政策大小限制。

日誌層級

您可以選擇 OFF、ALL、ERROR 或 FATAL。設定為 OFF 時不會記錄事件類型，設定為 ALL 時則會記錄所有事件類型。針對 ERROR 和 FATAL，請參閱下表。

如需有關針對「快速工作流程」執行所顯示之執行資料的詳細資訊記錄層級，請參閱主控台中的標準和快速工作流程執行。

事件類型	ALL	ERROR	FATAL	OFF
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
ExecutionAborted	✓	✓	✓	
ExecutionFailed	✓	✓	✓	
ExecutionStarted	✓			
ExecutionSucceeded	✓			
ExecutionTimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			

事件類型	ALL	ERROR	FATAL	OFF
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		
MapIterationFailed	✓	✓		
MapIterationStarted	✓			
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			

事件類型	ALL	ERROR	FATAL	OFF
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			
ParallelStateFailed	✓	✓		
ParallelStateStarted	✓			
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			

事件類型	ALL	ERROR	FATAL	OFF
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		
TaskState Aborted	✓	✓		
TaskState Entered	✓			
TaskStateExited	✓			
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			
TaskTimedOut	✓	✓		
WaitState Aborted	✓	✓		
WaitState Entered	✓			
WaitStateExited	✓			

AWS X-Ray 和 Step Functions

您可以用AWS X-Ray來視覺化狀態機器的元件、識別效能瓶頸，以及疑難排解導致錯誤的要求。您的狀態機會將追蹤資料傳送至 X-Ray，X-Ray 會處理資料以產生服務對應和可搜尋的追蹤摘要。

啟用狀態機的 X-Ray 後，您可以在所有可用 X-Ray 的 AWS 區域中，在 Step Functions 中執行時追蹤要求。這為您提供了整個 Step Functions 請求的詳細概述。即使追蹤 ID 未由上游服務傳遞，Step Functions 也會將追蹤傳送至 X-Ray 以執行狀態機器。您可以使用 X-Ray 服務對應來檢視要求的延遲時間，包括與 X-Ray 整合的任何 AWS 服務。您也可以設定取樣規則，根據您指定的準則告知 X-Ray 要記錄哪些請求，以及採樣率。

當您的狀態機器未啟用 X-Ray，且上游服務未傳遞追蹤 ID 時，Step Functions 不會將追蹤傳送至 X-Ray 以執行狀態機器。但是，如果追蹤 ID 是由上游服務傳遞的，Step Functions 就會將追蹤傳送至 X-Ray 以執行狀態機器。

您可以在兩者皆受支援的區域中 AWS X-Ray 搭配使用 Step Functions。請參閱步驟函數和 X-Ray 端點和配額頁面，以取得有關 X-Ray 和 Step Functions 區域支援的資訊。

> **X-Ray 和 Step Functions 合併配額**
>
> 您可以將資料新增至追蹤最多七天，並查詢追蹤資料可追溯至三十天，也就是 X-Ray 儲存追蹤資料的時間長度。您的痕跡將受到 X-Ray 配額的限制。除了其他配額外，X-Ray 還為 Step Functions 狀態機提供 100KB 的最小保證跟踪大小。如果將超過 100 KB 的追蹤資料提供給 X-Ray，這可能會導致軌跡凍結。如需其他 X-Ray 配額的詳細資訊，請參閱 X-Ray 端點和配額頁面的服務配額一節。

> **Important**
>
> Step Functions 不支援由「分散式地圖」狀態開始的子工作流程執行的 X-Ray 追蹤，因為很容易超過此類執行的「追蹤」文件大小限制。

主題

- 設置和配置
- 概念
- Step Functions 服務集成和 X-Ray
- 檢視 X-Ray 主控台
- 檢視 Step Functions 的 X-Ray 追蹤資訊
- 追蹤
- 服務地圖

- 區段和子區段
- 分析
- 組態
- 如果跟踪圖或服務地圖中沒有數據怎麼辦？

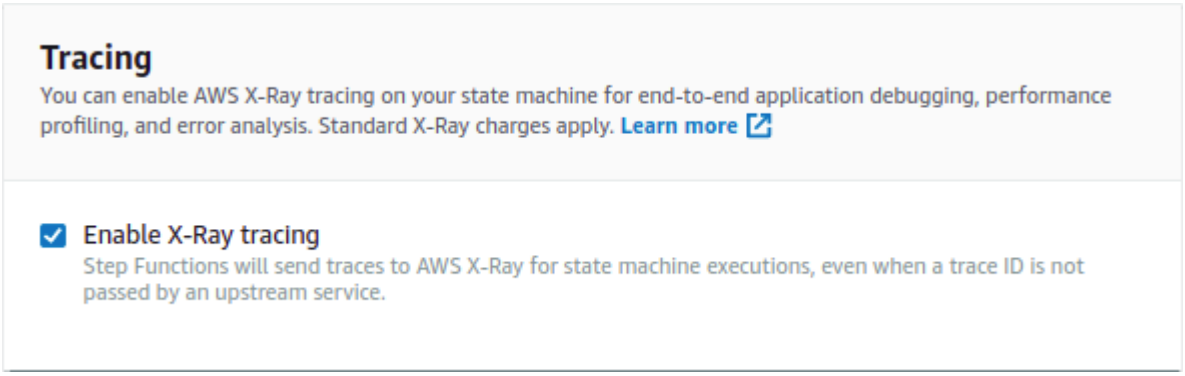
設置和配置

建立狀態機時啟用 X-Ray 追蹤

您可以在建立新狀態機時啟用 X-Ray 追蹤，方法是選取 [指定詳細資訊] 頁面上的啟用 X-Ray 追蹤。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在 [選擇編寫方法] 頁面上，選擇適當的選項來建立您的狀態機器。如果選擇「執行範例專案」，則無法在建立狀態機期間啟用 X-Ray 追蹤，並且在建立狀態機之後，您將需要啟用 X-Ray 追蹤。若要取得有關在現有狀態機中啟用 X-Ray 的更多資訊，請參閱在現有狀態機中啟用 X-Ray。

   選擇下一步。
3. 在 [指定詳細資料] 頁面上，設定您的狀態機器。
4. 選擇「啟動 X-Ray 追蹤」。



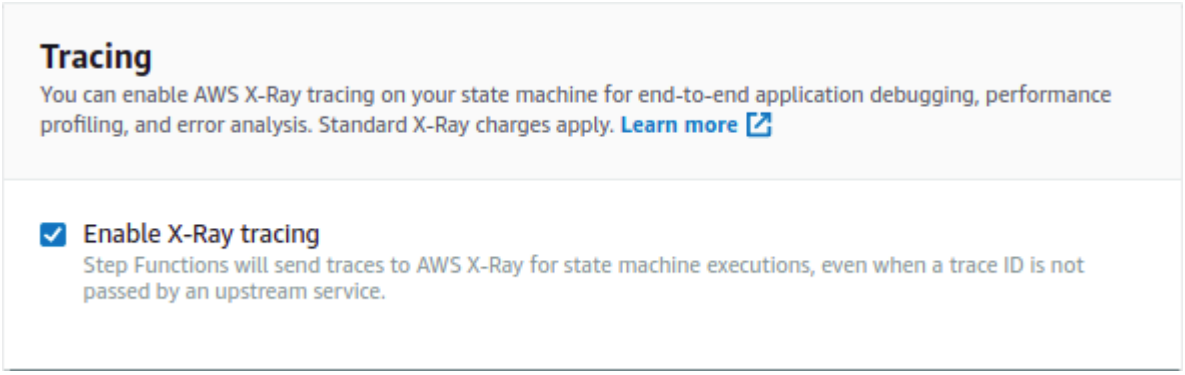
您的 Step Functions 狀態機現在將向 X-Ray 發送跟踪以執行狀態機。

> **Note**
>
> 如果您選擇使用現有的 IAM 角色，則應確保允許 X-Ray 寫入。如需有關所需許可的詳細資訊，請參閱 X-Ray 的 IAM 政策。

在現有狀態機中啟用 X-Ray

若要在現有狀態機中啟用 X-Ray：

1. 在 Step Functions 主控台中，選取要啟用追蹤的狀態機器。
2. 選擇編輯。
3. 選擇「啟動 X-Ray 追蹤」。



您將看到一條通知，告訴您可能需要進行其他更改。

> **Note**
>
> 當您為現有狀態機器啟用 X-Ray 時，您必須確保擁有可授予 X-Ray 足夠許可的 IAM 政策來執行追蹤。您可以手動新增一個，也可以產生一個。如需詳細資訊，請參閱的 IAM 政策一節的 IAM 政策 AWS X-Ray。

4. (選擇性) 為狀態機自動產生新角色，以包含 X-Ray 權限。
5. 選擇儲存。

設定 Step Functions 的 X-Ray 追蹤

當您第一次在啟用 X-Ray 追蹤的情況下執行狀態機時，它將使用 X-Ray 追蹤的預設組態值。 AWS X-Ray 不會針對傳送至應用程式的每個要求收集資料。相反，它會針對統計上顯著數量的請求收集資料。預設值是每秒記錄第一個要求，以及任何其他要求的百分之五。每秒一個請求是儲槽。這可確保只要服務持續提供請求，每秒都會記錄至少一個追蹤。5% 是超過儲槽大小的額外請求抽樣「速率」。

為了避免開始使用時產生的服務費用，預設的抽樣費率都很保守。您可以設定 X-Ray 來修改預設取樣規則，並設定根據服務或要求內容套用取樣的其他規則。

例如，您可能想要停用取樣，並追蹤修改狀態或處理碼 AWS 帳戶 或交易之呼叫的所有要求。對於大容量的唯讀呼叫 (例如背景輪詢、健康狀態檢查或連線維護)，您可以以低速率取樣，仍然可以取得足夠的資料來觀察發生的問題。

若要設定狀態機的取樣規則：

1. 轉到 X-Ray 控制台。
2. 選擇抽樣。
3. 若要建立規則，請選擇 Create sampling rule (建立抽樣規則)。

   若要編輯規則，請選擇規則的名稱。

   若要刪除規則，請選擇規則並使用 Actions (動作) 功能表來刪除它。

現有抽樣規則的某些部分，例如名稱和優先順序，無法變更。請改為新增或複製現有規則，進行所需的變更，然後使用新規則。

如需 X-Ray 取樣規則以及如何設定各種參數的詳細資訊，請參閱在 X-Ray 主控台中設定取樣規則。

整合上游服務

若要整合 Step Functions 工作流程的執行 (例如快速、同步和標準工作流程) 與上游服務，您需要設定`traceHeader`. 如果您在 API Gateway 中使用 HTTP API，系統會自動為您完成此操作。但是，如果您使用的是 Lambda 函數和/或 SDK，則需要自`traceHeader`行設置`StartExecution`或`StartSyncExecution`API 調用。

您必須將`traceHeader`格式指定為`\p{ASCII}#`。此外，若要讓 Step Functions 式使用相同的追蹤識別碼，您必須將格式指定為`Root={TRACE_ID};Sampled={1 or 0}`。如果您使用的是 Lambda 函數，請以目前區段中`TRACE_ID`的追蹤 ID 取代，並將「取樣」欄位設定為「取樣」欄位，`1`就好像您的取樣模式為 true，且`0`取樣模式為 false。提供此格式的追蹤 ID 可確保您取得完整的追蹤。

以下是用 Python 編寫的示例，以展示如何指定`traceHeader`.

```
state_machine = config.get_string_paramter("STATE_MACHINE_ARN")
    if (xray_recorder.current_subsegment() is not None and
        xray_recorder.current_subsegment().sampled) :
        trace_id = "Root={};Sampled=1".format(
            xray_recorder.current_subsegment().trace_id
        )
```

```
    else:
        trace_id = "Root=not enabled;Sampled=0"
    LOGGER.info("trace %s", trace_id)

    # execute it
    response = states.start_sync_execution(
        stateMachineArn=state_machine,
        input=event['body'],
        name=context.aws_request_id,
        traceHeader=trace_id
    )
    LOGGER.info(response)
```

概念

X-Ray 控制台

主 AWS X-Ray 控台可讓您檢視應用程式所提供之要求的服務對應和追蹤。您可以存取主控台，以檢視狀態機啟用 X-Ray 收集的詳細資訊。

如需有檢視 X-Ray 主控台關如何存取狀態機執行的 X-Ray 主控台的資訊，請參閱。

如需 X-Ray 主控台的詳細資訊，請參閱 X-Ray 主控台文件。

區段、子區段和軌跡

區段會記錄您狀態機器的請求相關資訊。它包含諸如狀態機執行的工作之類的信息，還可能包含有關下游呼叫的信息的子段。

追蹤會收集單一要求所產生的所有區段。

抽樣

為了確保有效追蹤，並提供應用程式所提供之請求的代表性樣本，X-Ray 會套用取樣演算法來判斷要追蹤哪些要求。這可以透過編輯取樣規則來變更。

指標

對於您的狀態機，X-Ray 將計量調用時間，狀態轉換時間，Step Functions 的總執行時間以及此執行時間內的變化。此資訊可透過 X-Ray 主控台存取。

分析

AWS X-Ray Analytics 主控台是一種用於解譯追蹤資料的互動式工具。您可以按一下與目前追蹤集相關聯之指標和欄位的圖表和面板，使用愈益精細的篩選條件來精簡作用中的資料集。這可讓您分析狀態機的執行情況，並快速找出並識別效能問題。

如需有關 X-Ray 分析的詳細資訊，請參閱與 AWS X-Ray Analytics 主控台互動

Step Functions 服務集成和 X-Ray

與 Step Functions 整合的某些 AWS 服務提供整合 AWS X-Ray ，方法是將追蹤標頭新增至要求、執行 X-Ray 精靈，或是做出取樣決策，並將追蹤資料上傳至 X-Ray。其他人必須使用 AWS X-Ray SDK 進行檢測。一些尚未支持 X-Ray 集成。使用與 Step Functions 數的服務整合時，X-Ray 整合是必要的，以提供完整的追蹤資料

原生 X-Ray 支援

與原生 X-Ray 支援的服務整合包括：

- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS Lambda
- AWS Step Functions

所需儀器

需要 X-Ray 儀器的服務集成：

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

僅用戶端追蹤

其他服務整合不支援 X-Ray 追蹤。但是，仍然可以收集客戶端跟踪：

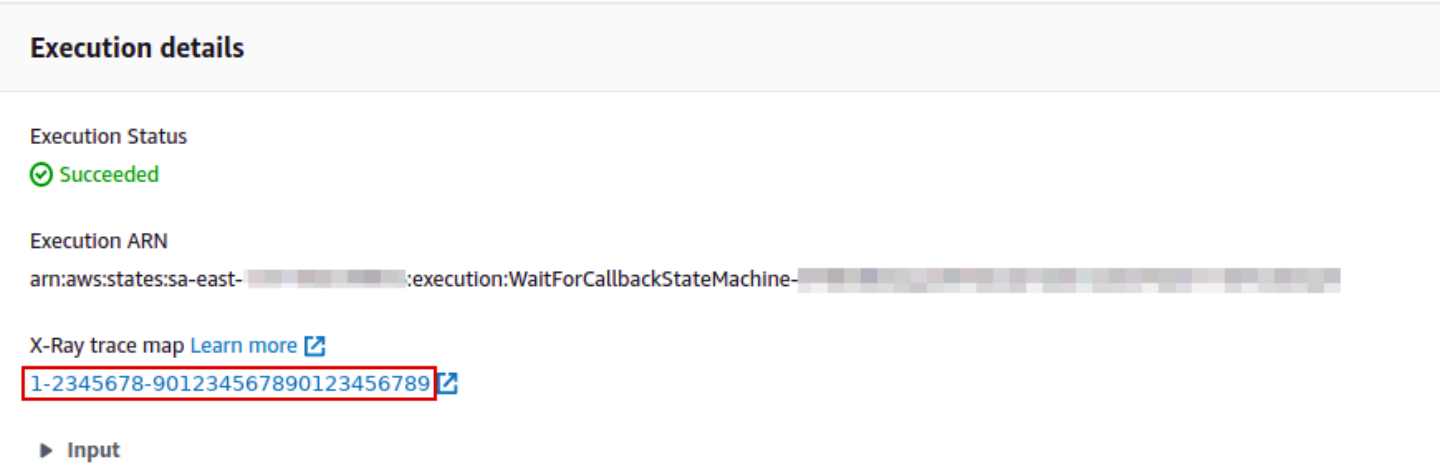
- Amazon DynamoDB

- Amazon EMR
- Amazon SageMaker
- AWS CodeBuild
- AWS Glue

檢視 X-Ray 主控台

X-Ray 接收來自服務的資料作為區段。X-Ray 會將具有共同要求的區段群組為追蹤。X-Ray 會處理追蹤，以產生提供應用程式視覺化呈現的服務圖表。

開始執行狀態機後，您可以在「執行詳細資訊」區段中選擇 X-Ray 追蹤對應連結來檢視其 X-Ray 軌跡。



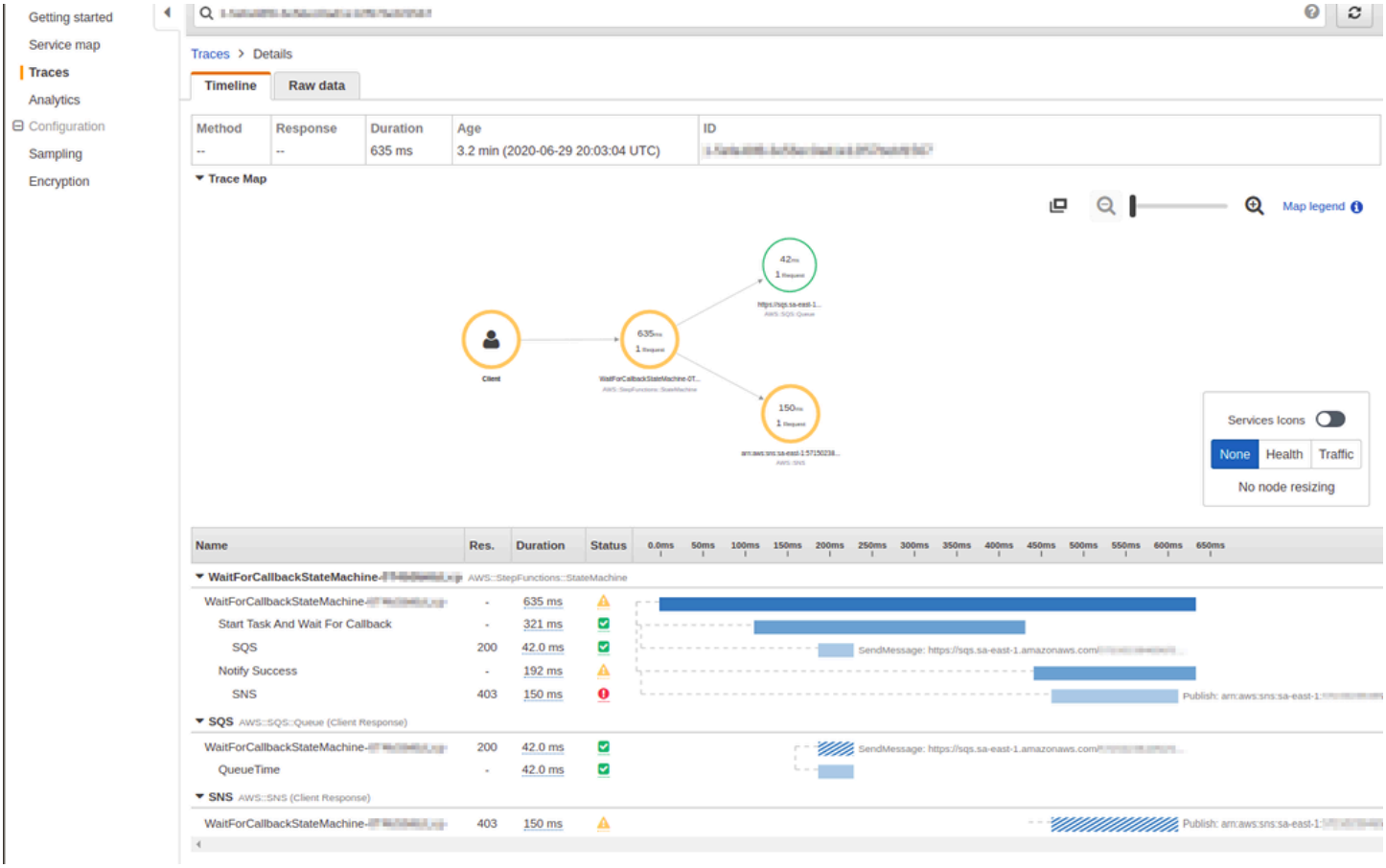
為狀態機啟用 X-Ray 之後，您可以在 X-Ray 主控台中檢視其執行的追蹤資訊。

檢視 Step Functions 的 X-Ray 追蹤資訊

下列步驟說明啟用 X-Ray 並執行執行後，您可以在主控台中看到的資訊類型。會展示回呼模式範例 (Amazon SQS、Amazon SNS、Lambda) 樣本專案的 X-Ray 軌跡。

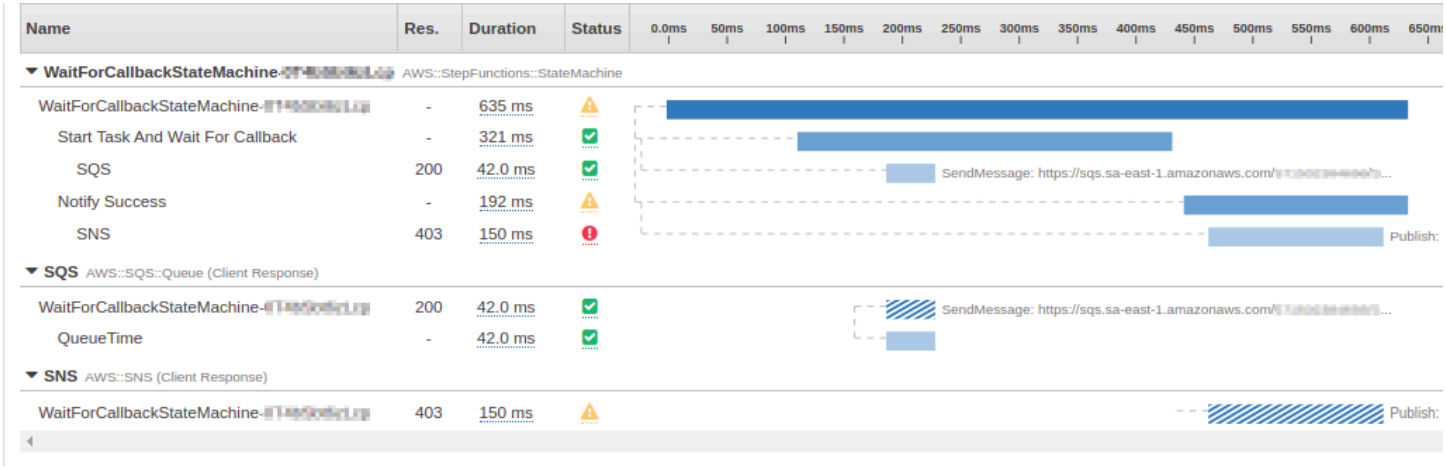
追蹤

執行完成後，您可以瀏覽至 X-Ray 主控台，您將在其中看到「X-Ray 追蹤」頁面。這會顯示服務對應的概觀，以及狀態機器的追蹤和區段資訊。

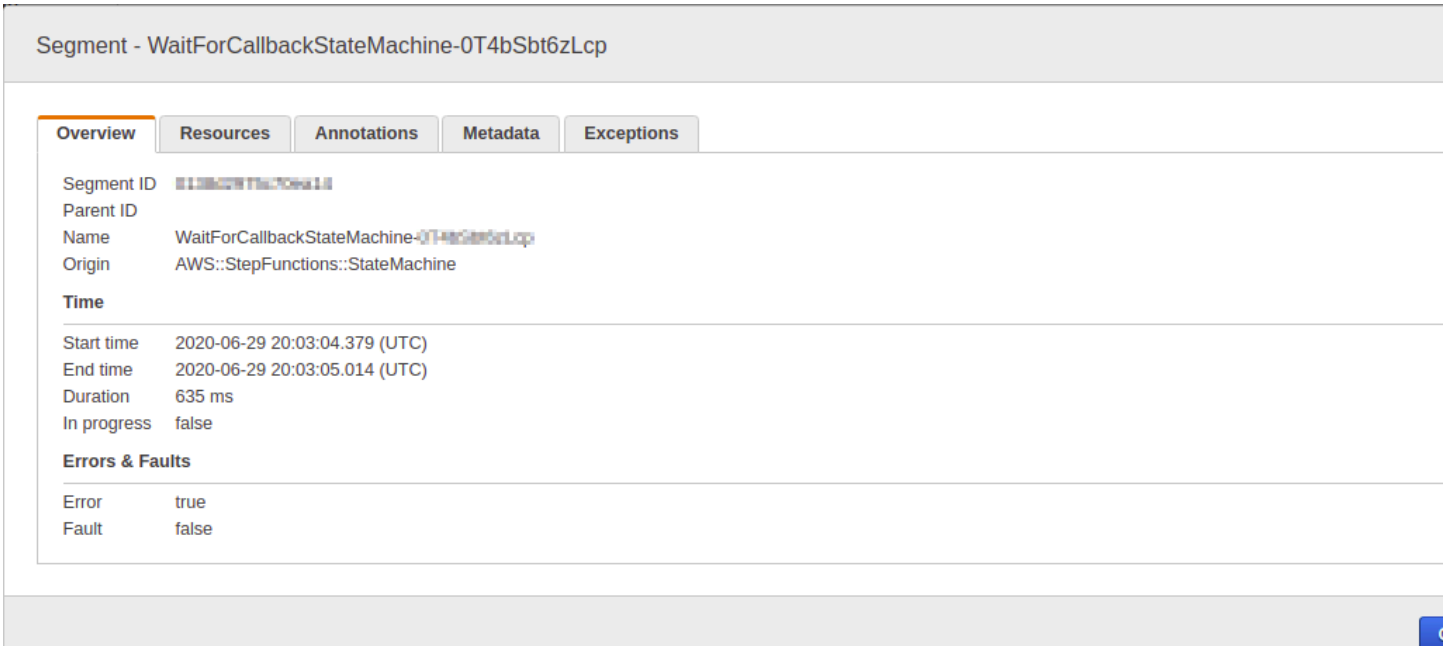


服務地圖

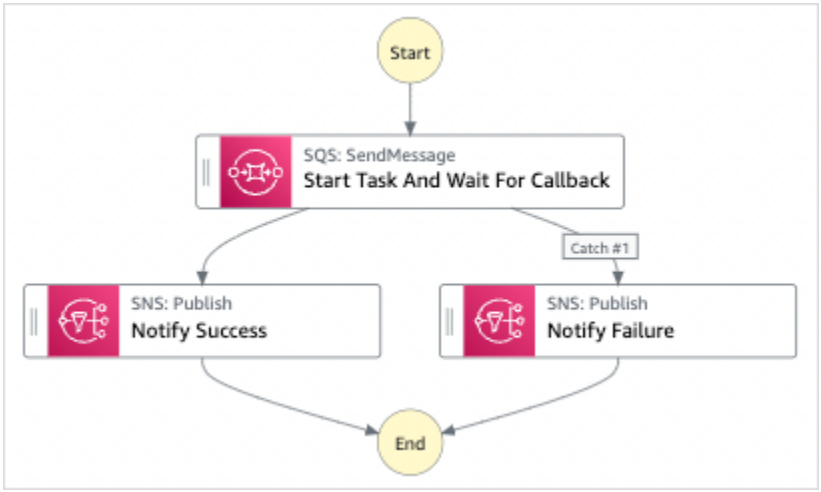
X-Ray 主控台中的服務對應可協助您識別發生錯誤的服務、有高延遲連線的服務，或查看未成功要求的追蹤。



在追蹤對映上，您可以選擇服務節點來檢視該節點的要求，或選擇兩個節點之間的邊緣來檢視傳遞該連線的要求。您可以在此選擇`WaitForCallBack`節點，并查看有關其執行狀態和響應狀態的其他信息。



您可以看到 X-Ray 服務地圖與狀態機的關聯性。如果 Step Functions 支援 X-Ray，則每個服務整合都有一個服務對應節點。



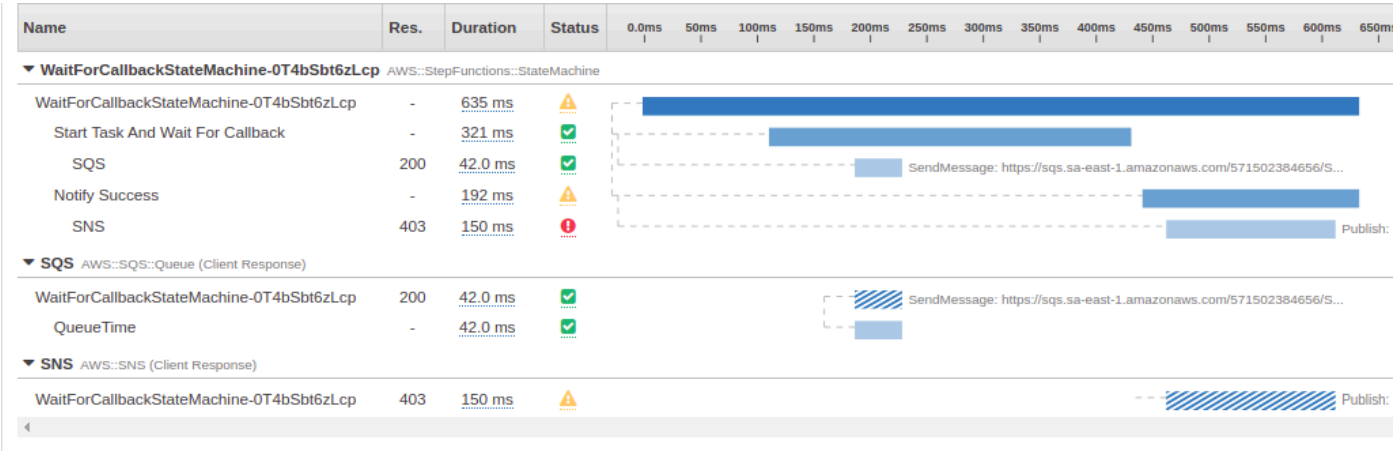
區段和子區段

追蹤是單一要求所產生的區段集合。每個區段都會提供資源的名稱、有關請求的詳細資訊，以及已完成工作的詳細資訊。在「追蹤」頁面上，您可以看到區段，如果展開，則會看到其對應的子區段。您可以選擇段或子段來檢視有關該段或子段的詳細資訊。

選擇每個標籤，以查看區段和子區段資訊的顯示方式。

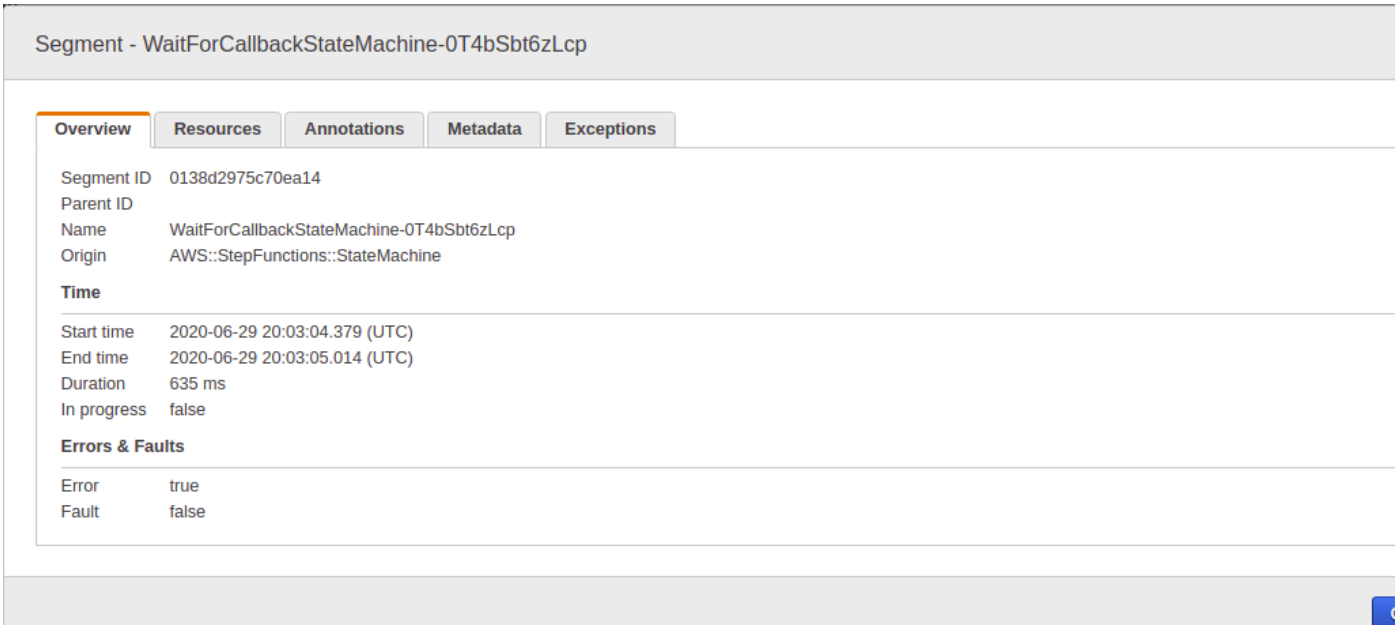
Overview of Segments

此狀態機器的區段和子區段概觀。服務對應上的每個節點都有不同的區段。



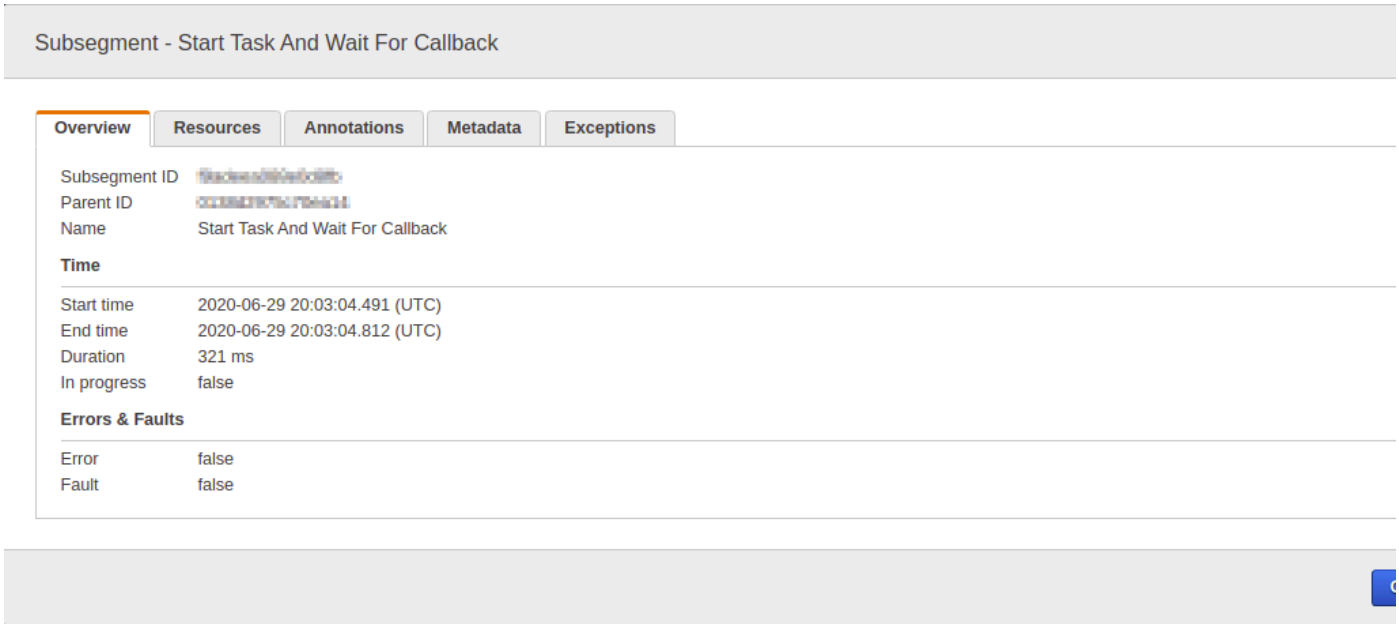
View segment detail

選擇區段可提供資源的名稱、有關請求的詳細資訊，以及已完成工作的詳細資訊。



View subsegment detail

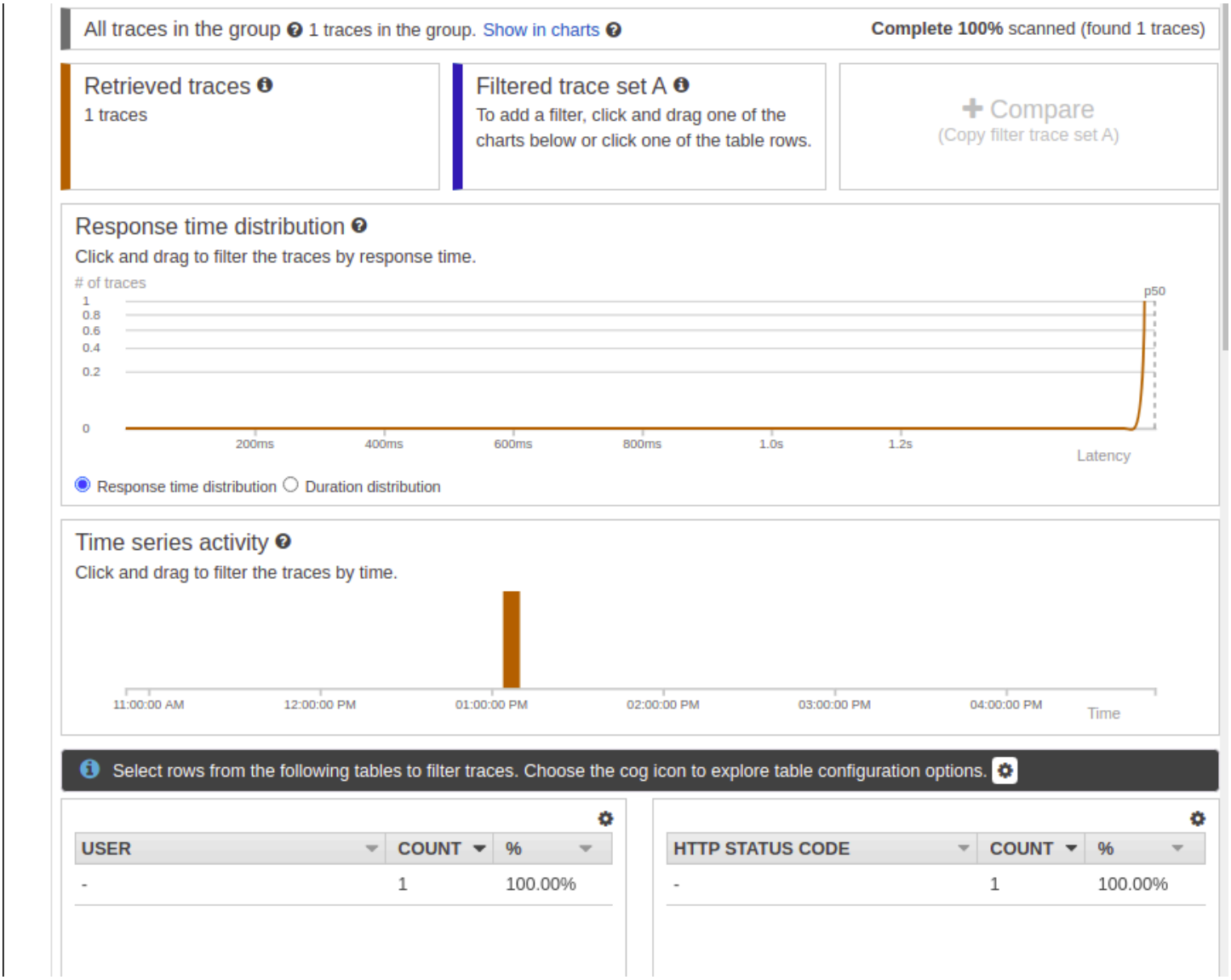
區段可以將完成工作的資料細分為子區段。選擇子區段可讓您檢視更精細的時間資訊和詳細資訊。子區段可包含有關呼叫 AWS 服務、外部 HTTP API 或 SQL 資料庫的其他詳細資料。



分析

AWS X-Ray Analytics 主控台是一種用於解譯追蹤資料的互動式工具。您可以使用它來更輕鬆地了解狀態機的執行情況。主控台可讓您透過互動的回應時間和時間序列圖表，探索、分析和視覺化追蹤。這可協助您快速找出效能和延遲問題。

您可以按一下與目前追蹤集相關聯之指標和欄位的圖表和面板，使用愈益精細的篩選條件來精簡作用中的資料集。

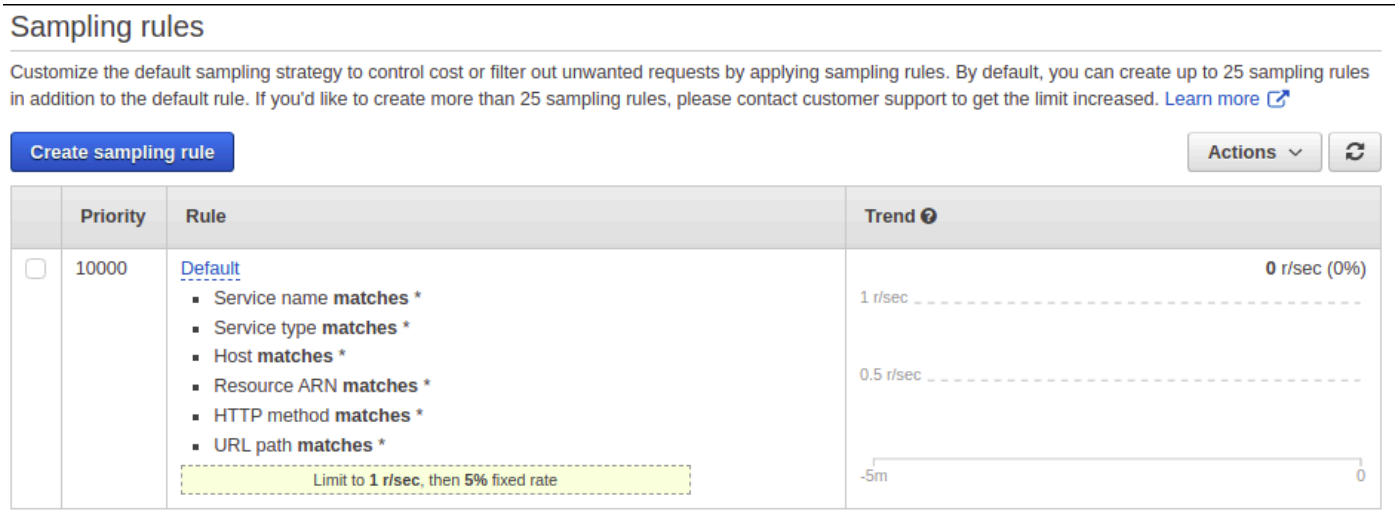


組態

您可以從 X-Ray 主控台設定取樣和加密選項。

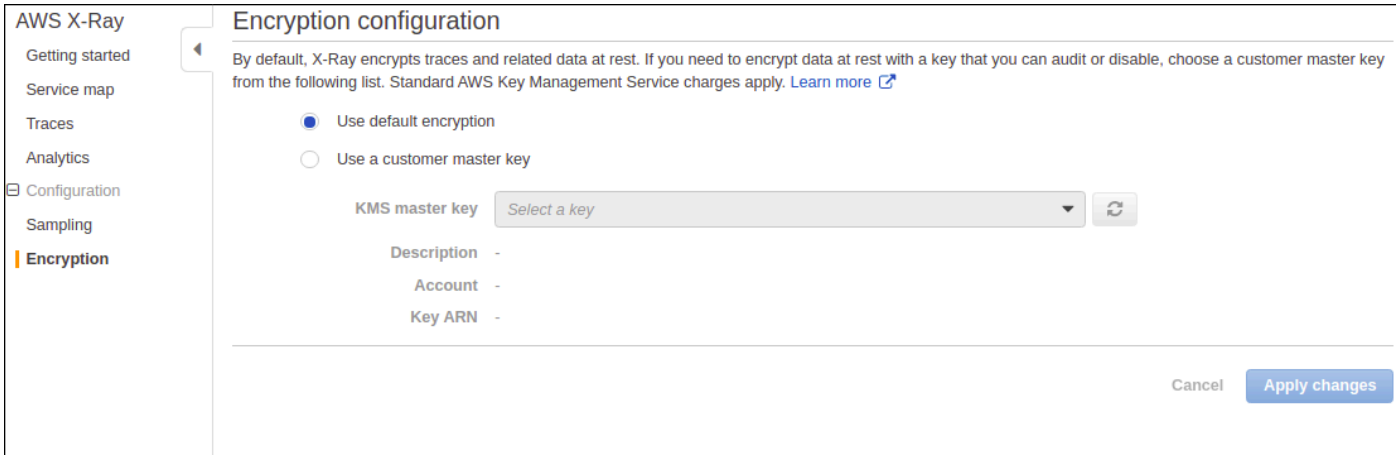
Sampling

選擇取樣以檢視有關取樣率和組態的詳細資訊。您可以變更取樣規則以控制記錄的資料量，並修改取樣行為以符合您的特定需求。



Encryption

選擇加密以修改加密設定。您可以使用預設設定，X-Ray 會加密追蹤和靜止日期，或者，如果需要，您可以選擇客戶主金鑰。標準AWS KMS費用適用於後一種情況。



如果跟踪圖或服務地圖中沒有數據怎麼辦？

如果您已啟用 X-Ray，但在 X-Ray 主控台中看不到任何資料，請檢查：

- 您的 IAM 角色設定正確，以允許寫入 X-Ray。
- 採樣規則允許數據的採樣。
- 由於在套用新建立或修改的 IAM 角色之前可能會有短暫的延遲，因此請在幾分鐘後再次檢查追蹤或服務對應。

- 如果您在「X-Ray 追蹤」面板中看到「找不到資料」，請檢查您的 IAM 帳戶設定，並確定 AWS Security Token Service 已針對預定區域啟用。如需詳細資訊，請參閱《IAM 使用者指南》AWS STSAWS 區域中的「啟用和停用」。

搭配使用 AWS 使用者通知 與 AWS Step Functions

您可以用AWS 使用者通知來設定傳送管道，以接收有關AWS Step Functions事件的通知。當事件符合您指定的規則時，便會收到通知。您可以透過多個管道接收事件通知，包括電子郵件、AWS Chatbot 聊天通知或 AWS Console Mobile Application 推送通知。您也可在 Console Notifications Center 中看到通知。使用者通知 支援彙總，可減少您在特定事件期間收到的通知數目。

中的安全性 AWS Step Functions

本節提供有關 AWS Step Functions 安全性和驗證的資訊。

主題

Step Functions 使用 IAM 來控制對其他 AWS 服務和資源的存取。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。

資料保護 AWS Step Functions

AWS 共用責任模型適用於中的資料保護 AWS Step Functions。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。

- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 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，例如名稱欄位。這包括當您使用主控台、API 或 AWS SDK AWS 服務 使用 Step Functions 或其他功能時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

中的加密 AWS Step Functions

靜態加密

Step Functions 數一律會加密您的靜態資料。中的 AWS Step Functions 資料會使用透明伺服器端加密進行靜態加密。這可協助降低保護敏感資料所涉及的操作負擔和複雜性。您可以透過靜態加密，建立符合加密合規和法規要求，而且對安全性要求甚高的應用程式

傳輸中加密

Step Functions 會加密服務與其他整合式 AWS 服務之間傳輸中的資料 (請參閱AWS Step Functions 搭配其他服務使用)。在 Step Functions 數和整合式服務之間傳遞的所有資料都會使用傳輸層安全性 (TLS) 加密。

AWS Step Functions中的 Identity and Access Management

存取權 AWS Step Functions 需要 AWS 可用來驗證您的請求的憑證。這些認證必須具有存取 AWS 資源的權限，例如從其他 AWS 資源擷取事件資料。以下各節詳細說明如何使用 AWS Identity and Access Management (IAM) 和 Step Functions，藉由控制可存取資源的使用者來協助保護資源的安全。

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

物件

您如何使用 AWS Identity and Access Management (IAM) 會有所不同，這取決於您在 Step Functions 中執行的工作。

服務使用者 — 如果您使用 Step Functions 服務來執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 Step Functions 能來完成工作時，您可能需要其他權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Step Functions 中的特徵，請參閱疑難排解 AWS Step Functions 身分和存取。

服務管理員 — 如果您負責公司的 Step Functions 資源，您可能擁有對 Step Functions 的完整存取權。決定您的服務使用者應該存取哪些 Step Functions 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要深入瞭解貴公司如何搭配 Step Functions 使用 IAM，請參閱如何與 IAM AWS Step Functions 搭配使用。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要瞭解如何撰寫政策以管理 Step Functions 存取權限的詳細資訊。若要檢視可在 IAM 中使用的 Step Functions 以身分識別為基礎的政策範例，請參閱。以身分識別為基礎的原則範例 AWS Step Functions

使用身分驗證

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

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

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

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

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入

來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

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

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

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

使用政策管理存取權

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

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

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

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

存取控制

您可以擁有有效的認證來驗證您的請求，但除非您擁有權限，否則您無法建立或存取 Step Functions 資源。例如，您必須具有與 Step Functions 規則關聯的叫用 AWS Lambda、Amazon 簡單通知服務 (Amazon SNS) 和 Amazon 簡單佇列服務 (Amazon SQS) 目標的許可。

以下各節說明如何管理 Step Functions 的權限。

- 為狀態機建立 IAM 角色
- 為非管理員使用者建立精細的 IAM 許可
- 適用於 Step Functions 的 Amazon VPC 端點
- 整合式服務的 IAM 政策
- 使用分散式地圖狀態的 IAM 政策

Step Functions 的原則動作

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Step Functions 動作清單，請參閱服務授權參考 AWS Step Functions中的定義資源。

Step Functions 中的原則動作會在動作之前使用下列前置詞：

```
states
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "states:action1",
      "states:action2"
         ]
```

若要檢視 Step Functions 以身分識別為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Step Functions

Step Functions 的原則資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 Step Fun ctions 資源類型及其 ARN 的清單，請參閱服務授權參考 AWS Step Functions中的定義動作。若要瞭解您可以使用哪些動作指定每個資源的 ARN，請參閱資源定義者 AWS Step Functions。

若要檢視 Step Functions 以身分識別為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Step Functions

Step Functions 的原則條件索引鍵

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看 Step Functions 條件索引鍵的清單，請參閱服務授權參考 AWS Step Functions中的條件索引鍵。若要瞭解您可以使用條件索引鍵的動作和資源，請參閱資源定義者 AWS Step Functions。

若要檢視 Step Functions 以身分識別為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Step Functions

Step Functions 中的 ACL

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

具有 Step Functions 的 ABAC

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

搭配 Step Functions 使用臨時認證

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Step Functions 的跨服務主體權限

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Step Functions 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Warning**
>
> 變更服務角色的權限可能會中斷 Step Functions 功能。只有當 Step Functions 提供指引時，才編輯服務角色。

Step Functions 的服務連結角色

支援服務連結角色。	否

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 `Yes`。選擇是連結，以檢視該服務的服務連結角色文件。

如何與 IAM AWS Step Functions 搭配使用

在您使用 IAM 管理 Step Functions 的存取權限之前，請先了解哪些 IAM 功能可與 Step Functions 搭配使用。

您可以搭配使用的 IAM 功能 AWS Step Functions

IAM 功能	Step Functions 支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分

IAM 功能	Step Functions 支援
臨時憑證	是
主體許可	是
服務角色	是
服務連結角色	否

若要深入瞭解 Step Functions 式和其他 AWS 服務如何與大多數 IAM 功能搭配運作，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

以身分識別為基礎的原則範例 AWS Step Functions

根據預設，使用者和角色沒有建立或修改 Step Functions 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需有關 Step Functions 定義之動作和資源類型的詳細資訊，包括每個資源類型的 ARN 格式，請參閱服務授權參考 AWS Step Functions中的動作、資源和條件索引鍵。

主題

- 政策最佳實務
- 使用 Step Functions 主控台
- 允許使用者檢視他們自己的許可

政策最佳實務

以身分識別為基礎的政策會決定使用者是否可以在您的帳戶中建立、存取或刪除 Step Functions 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 Step Functions 主控台

若要存取 AWS Step Functions 主控台，您必須擁有最少一組權限。這些權限必須允許您列出和檢視有關 AWS 帳戶. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

若要確保使用者和角色仍可使用 Step Functions 主控台，請同時將 Step Functions *ConsoleAccess* 或*ReadOnly* AWS 受管理的原則附加至實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Step Functions 的識別型原則

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Step Functions 的識別型原則範例

若要檢視 Step Functions 以身分識別為基礎的原則範例，請參閱。以身分識別為基礎的原則範例 AWS Step Functions

Step Functions 中的資源型政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

為狀態機建立 IAM 角色

AWS Step Functions 可以執行代碼和訪問 AWS 資源（例如調用 AWS Lambda 函數）。為了維護安全性，您必須使用 IAM 角色將這些資源的存取授予 Step Functions。

本指南Step Functions 的教學課程中的可讓您利用自動產生的 IAM 角色，這些角色對您建立狀態機的 AWS 區域有效。不過，您可以為狀態機器建立自己的 IAM 角色。

建立要使用的狀態機器的 IAM 政策時，該政策應包含您希望狀態機器承擔的許可。您可以使用現有的 AWS 受管理原則做為範例，也可以從頭開始建立符合您特定需求的自訂原則。如需詳細資訊，請參閱 IAM 使用者指南中的建立 IAM 政策

若要為狀態機建立自己的 IAM 角色，請遵循本節中的步驟。

在此範例中，您建立具有叫用 Lambda 函數之權限的 IAM 角色。

建立 Step Functions 的角色

1. 登入 IAM 主控台，然後選擇 [角色]、[建立角色]。
2. 在 [選取信任的實體] 頁面的 [AWS 服務] 下，從清單中選取 [Step Functions]，然後選擇 [下一步：權限]。
3. 在 Attached permissions policy (連結許可政策) 頁面上，選擇 Next: Review (下一步：檢閱)。
4. 在 [複查] 頁面上，輸入`StepFunctionsLambdaRole`角色名稱，然後選擇 [建立角色]。

   IAM 角色會顯示在角色清單中。

如需 IAM 許可和政策的詳細資訊，請參閱 IAM 使用者指南中的存取管理。

防止跨服務混淆副問題

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。在中 AWS，跨服務模擬可能會導致混淆的副問題。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。這種類型的冒充可能發生跨帳戶和跨服務。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。

為了防止代表混淆， AWS 提供了一些工具，協助您透過已授予您帳戶中資源存取權的服務主體來保護所有服務的資料。本節著重於特定的跨服務混淆副預防 AWS Step Functions；不過，您可以在 IAM 使用者指南中混淆的副問題一節中深入了解此主題。

我們建議在資源策略中使用`aws:SourceArn`和`aws:SourceAccount`全域條件內容索引鍵，以限制授Step Functions予其他服務存取資源的權限。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。如果您不知道資源的完整 ARN，或者您要指定多個資源，請針對 ARN 的未知部分使用萬用字元 (*) 的`aws:SourceArn`全域內容條件索引鍵。例如 `arn:aws:states:*:111122223333:*`。

以下是受信任原則的範例，顯示如何使`aws:SourceAccount`用 `aws:SourceArn` Step Functions，以防止混淆的副問題。

```
{
```

```
  "Version":"2012-10-17",
  "Statement":[
     {
        "Effect":"Allow",
        "Principal":{
           "Service":[
              "states.amazonaws.com"
           ]
        },
        "Action":"sts:AssumeRole",
        "Condition":{
           "ArnLike":{
              "aws:SourceArn":"arn:aws:states:us-east-1:111122223333:stateMachine:*"
           },
           "StringEquals":{
              "aws:SourceAccount":"111122223333"
           }
        }
     }
  ]
}
```

連接內嵌政策

Step Functions 可以直接在Task狀態下控制其他服務。附加內嵌政策以允許 Step Functions 存取您需要控制之服務的 API 動作。

1. 開啟 IAM 主控台、選擇 [角色]、搜尋您的 Step Functions 角色，然後選取該角色。
2. 選取 Add inline policy (新增內嵌政策)。
3. 使用 Visual editor (視覺編輯工具) 或 JSON 標籤來建立角色的政策。

如需如何 AWS Step Functions 控制其他 AWS 服務的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

> **Note**
>
> 如需由 Step Functions 主控台建立的 IAM 政策範例，請參閱整合式服務的 IAM 政策。

為非管理員使用者建立精細的 IAM 許可

IAM 中的預設受管政策，例如`ReadOnly`，並不完全涵蓋所有類型的 AWS Step Functions 許可。本節描述不同類型的許可，並提供一些範例組態。

Step Functions 有四種權限類別。視您想要提供給使用者的存取權而定，您可以使用這些類別中的許可來控制存取權。

[服務層級許可]

適用於不會對特定資源採取行動的 API 元件。

[狀態機器層級許可]

適用於會對特定狀態機器採取行動的所有 API 元件。

[執行層級許可]

套用到會對特定執行作業採取行動的所有 API 元件。

[活動層級許可]

適用於會對特定活動或對活動的特定執行個體採取行動的所有 API 元件。

服務層級許可

此許可層級適用於不會對特定資源採取行動的所有 API 動作。其中包括 `CreateStateMachine`、`CreateActivity`、`ListStateMachines` 和 `ListActivities`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:ListActivities",
        "states:CreateStateMachine",
        "states:CreateActivity"
      ],
      "Resource": [
        "arn:aws:states:*:*:*"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam:::role/my-execution-role"
      ]
    }
  ]
}
```

狀態機器層級許可

此許可層級適用於會對特定狀態機器採取行動的所有 API 動作。這些 API 操作需要狀態機器的 Amazon 資源名稱 (ARN) 作為請求的一部分，例如DeleteStateMachineDescribeStateMachine、StartExecution、和ListExecutions。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeStateMachine",
        "states:StartExecution",
        "states:DeleteStateMachine",
        "states:ListExecutions",
        "states:UpdateStateMachine",
        "states:TestState",
        "states:RevealSecrets"
      ],
      "Resource": [
        "arn:aws:states:*:*:stateMachine:StateMachinePrefix*"
      ]
    }
  ]
}
```

執行層級許可

此許可層級適用於會對特定執行作業採取行動的所有 API 動作。這些 API 操作需要執行的 ARN 做為要求的一部分，例如 DescribeExecution、GetExecutionHistory 和 StopExecution。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
}
```

活動層級許可

此許可層級適用於會對特定活動或其特定執行個體採取行動的所有 API 動作。這些 API 操作需要活動的 ARN 或實例的令牌作為請求的一部分，例如DeleteActivity，DescribeActivityGetActivityTask，和SendTaskHeartbeat。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
```

```
      ]
    }
  ]
}
```

存取工作流程 AWS 帳戶 中其他資源

Step Functions 提供跨帳戶存取工作流程中不同設定 AWS 帳戶 的資源。使用 Step Functions 服務整合，即使 AWS 服務 不支援以 AWS 資源為基礎的政策或跨帳戶呼叫，您也可以叫用任何跨帳戶資源。

例如，假設你擁有兩個 AWS 帳戶，稱為開發和測試，在相同 AWS 區域。使用跨帳戶存取，開發帳戶中的工作流程可以存取測試帳戶中可用的資源，例如 Amazon S3 儲存貯體、Amazon DynamoDB 表和 Lambda 函數。

> **Important**
>
> IAM 角色和資源型政策只會在單一分割內跨帳戶委派存取許可。例如，假設您在標準 aws 分割區的美國西部 (加利佛尼亞北部) 中有一個帳戶。您在 aws-cn 分割區的中國 (北京) 中也有一個帳戶。您不能使用中國 (北京) 中帳戶的 Amazon S3 資源型政策，對標準 aws 帳戶中的使用者允許存取許可。

如需跨帳戶存取的詳細資訊，請參閱 IAM 使用者指南中的跨帳戶政策評估邏輯。

雖然每個 AWS 帳戶 都能完全控制自己的資源，但是使用 Step Functions，您可以重新組織、交換、新增或移除工作流程中的步驟，而不需要自訂任何程式碼。即使流程發生變化或應用程序的發展，您也可以執行此操作。

您還可以調用嵌套狀態機器的執行，以便在不同的帳戶中使用它們。這樣做可以有效地分離和隔離您的工作流程。當您在工作流程中使用.sync服務整合模式來存取其他帳戶中的其他 Step Functions 工作流程時，Step Functions 會使用耗用您指派配額的輪詢。如需詳細資訊，請參閱 執行任務 (.sync)。

> **Note**
>
> 目前，Step Functions 中無法使用跨區域 AWS SDK 整合和跨區域 AWS 資源存取。

目錄

- 呼叫跨帳戶資源
- 教學課程：存取跨帳戶 AWS 資源
- 跨帳戶存取 .sync 整合模式

本主題的關鍵概念

執行角色

Step Functions 數用來執行程式碼和存取 AWS 資源 (例如 AWS Lambda 函數的叫用動作) 的 IAM 角色。

服務整合

可從工作流程中的Task狀態中呼叫的 AWS SDK 整合 API 動作。

來源帳戶

擁 AWS 帳戶 有狀態機器並已開始執行的。

目標帳戶

您進行跨帳戶通話 AWS 帳戶 的目標。

目標角色

目標帳戶中的 IAM 角色，狀態機為對目標帳戶擁有的資源進行呼叫而假定。

執行 Job (.sync)

用於呼叫服務的服務整合模式，例如 AWS Batch. 它還使得 Step Functions 狀態機等待作業完成，然後再進入下一個狀態。若要指出 Step Functions 應該等待，請在Task狀態定義的Resource欄位中附加.sync尾碼。

呼叫跨帳戶資源

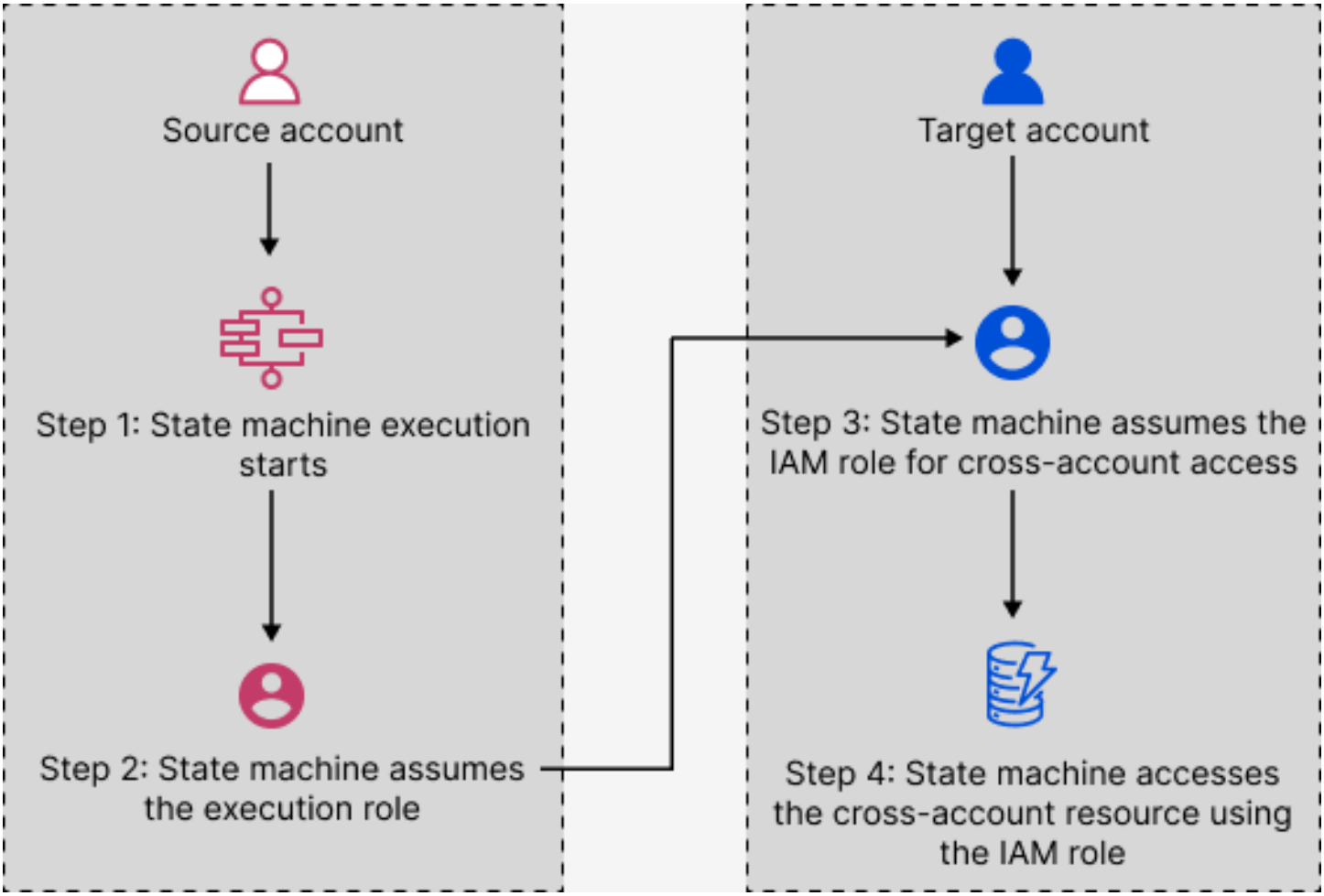
若要在工作流程中叫用跨帳號資源，請執行下列動作：

1. 在包含資源的目標帳戶中建立 IAM 角色。此角色會授與包含狀態機器的來源帳戶存取目標帳戶資源的權限。
2. 在Task州的定義中，指定狀態機器在叫用跨帳戶資源之前要採用的目標 IAM 角色。
3. 修改目標 IAM 角色中的信任政策，以允許來源帳戶暫時擔任此角色。信任政策必須包含來源帳戶中定義之狀態機器的 Amazon 資源名稱 (ARN)。此外，請在目標 IAM 角色中定義適當的許可，以呼叫 AWS 資源。

4. 更新來源帳戶的執行角色，以包含假設目標 IAM 角色所需的權限。

如需範例，請參閱教學課程：存取跨帳戶 AWS 資源。

> **Note**
>
> 您可以將狀態機設定為擔任 IAM 角色，以便從多個存取資源 AWS 帳戶。但是，狀態機器在給定時間只能承擔一個 IAM 角色。



教學課程：存取跨帳戶 AWS 資源

透過 Step Functions 中的跨帳戶存取支援，您可以共用不同 AWS 帳戶設定的資源。在本教學中，我們將引導您完成存取在名為生產的帳戶中定義的跨帳戶 Lambda 函數的程序。這個函數是從一個名為開發帳戶的狀態機調用。在本教學課程中，開發帳戶稱為來源帳戶，生產帳戶是包含目標 IAM 角色的目標帳戶。

首先，請在Task州的定義中指定狀態機器必須承擔的目標 IAM 角色，然後再叫用跨帳戶 Lambda 函數。然後，修改目標 IAM 角色中的信任政策，以允許來源帳戶暫時擔任目標角色。此外，若要呼叫資 AWS 源，請在目標 IAM 角色中定義適當的許可。最後，更新來源帳戶的執行角色，以指定承擔目標角色所需的權限。

您可以將狀態機設定為擔任 IAM 角色，以便從多個存取資源 AWS 帳戶。但是，根據州/省的定義，狀態機器在給定時間只能承擔一個 IAM 角色。Task

> **Note**
>
> 目前，Step Functions 中無法使用跨區域 AWS SDK 整合和跨區域 AWS 資源存取。

目錄

必要條件

- 本教學課程使用 Lambda 函數範例來示範如何設定跨帳戶存取。您可以使用任何其他 AWS 資源，但請確定您已在不同的帳號中設定資源。

  > **Important**
  >
  > IAM 角色和資源型政策只會在單一分割內跨帳戶委派存取許可。例如，假設您在標準 aws 分割區的美國西部 (加利佛尼亞北部) 中有一個帳戶。您在 aws-cn 分割區的中國 (北京) 中也有一個帳戶。您不能使用中國 (北京) 中帳戶的 Amazon S3 資源型政策，對標準 aws 帳戶中的使用者允許存取許可。

- 在文本文件中記下跨帳戶資源的 Amazon 資源名稱（ARN）。稍後在本教程中，您將在狀態機的Task狀態定義中提供此 ARN。以下是一個 Lambda 函數 ARN 的示例：

```
arn:aws:lambda:us-east-2:123456789012:function:functionName
```

- 確保您已創建狀態機需要承擔的目標 IAM 角色。

步驟 1：更新工作狀態定義以指定目標角色

在工作流程的`Task`狀態中，新增一個`Credentials`欄位，其中包含狀態機器在叫用跨帳戶 Lambda 函數之前必須承擔的身分。

下列程序示範如何存取名為的跨帳戶 Lambda 函`Echo`數。您可以按照以下步驟調用任何 AWS 資源。

1. 開啟 Step Functions 主控台，然後選擇建立狀態機器。
2. 在 [選擇編寫方法] 頁面上，選擇 [視覺化設計工作流程]，並保留所有預設選取項目。
3. 若要開啟「工作流程工作室」，請選擇
4. 在 [動作] 索引標籤上，將`Task`狀態拖放到畫布上。這會叫用正在使用此狀態的跨帳戶 Lambda 函數。`Task`
5. 在「組態」索引標籤上，執行下列動作：

   a. 將狀態重新命名為**`Cross-account call`**。

   b. 對於函數名稱，選擇輸入函數名稱，然後在方塊中輸入 Lambda 函數 ARN。例如`arn:aws:lambda:us-east-2:111122223333:function:Echo`。

   c. 對於提供身分與存取權管理角色 ARN，請指定目標 IAM 角色 ARN。例如`arn:aws:iam::111122223333:role/LambdaRole`。

   > **Tip**
   >
   > 或者，您也可以在包含 IAM 角色 ARN 的狀態 JSON 輸入中指定現有鍵值對的參考路徑。若要這麼做，請從狀態輸入中選擇 [在執行階段取得 IAM 角色 ARN]。如需使用參考路徑指定值的範例，請參閱將 JSON 路徑指定為 IAM 角色 ARN。

6. 選擇下一步。
7. 在 [檢閱產生的字碼] 頁上，選擇 [下一步]。
8. 在 [指定狀態機器設定] 頁面上，指定新狀態機器的詳細資料，例如名稱、權限和記錄層級。
9. 選擇 Create state machine (建立狀態機器)。
10. 在文字檔中記下狀態機器的 IAM 角色 ARN 和狀態機器 ARN。您需要在目標帳戶的信任策略中提供這些 ARN。

您的`Task`狀態定義現在看起來應該類似於下列定義。

```
{
  "StartAt": "Cross-account call",
  "States": {
    "Cross-account call": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
        "RoleArn": "arn:aws:iam::111122223333:role/LambdaRole"
      },
      "Parameters": {
        "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:Echo",
      },
      "End": true
    }
  }
}
```

步驟 2：更新目標角色的信任原則

IAM 角色必須存在於目標帳戶中，而且您必須修改其信任政策，以允許來源帳戶暫時擔任此角色。此外，您還可以控制誰可以擔任目標 IAM 角色。

建立信任關係之後，來源帳戶的使用者可以使用 AWS Security Token Service (AWS STS) AssumeRoleAPI 作業。此作業提供暫時的安全登入資料，可讓您存取目標帳戶中的 AWS 資源。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在主控台的導覽窗格中，選擇 [角色]，然後使用 [搜尋] 方塊搜尋目標 IAM 角色。例如 *LambdaRole*。
3. 選擇信任關係標籤。
4. 選擇 [編輯信任原則]，然後貼上下列信任原則。確保替換 AWS 帳戶 數字和 IAM 角色 ARN。該`sts:ExternalId`字段進一步控制誰可以擔任該角色。狀態機的名稱必須只包含 AWS Security Token Service `AssumeRole` API 支援的字元。如需詳細資訊，請參閱 AWS Security Token Service API 參考AssumeRole中的。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": "sts:AssumeRole",
   ```

```
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ExecutionRole"  // The source
 account's state machine execution role ARN
      },
      "Condition": {  // Control which account and state machine can assume the
 target IAM role

        "StringEquals": {
          "sts:ExternalId": "arn:aws:states:us-
east-1:123456789012:stateMachine:testCrossAccount"   //// ARN of the state machine
 that will assume the role.
        }
      }
    }
  ]
}
```

5. 保持此視窗開啟，並繼續執行下一個步驟以進行進一步的動作。

步驟 3：在目標角色中添加所需的權限

IAM 政策中的許可決定是允許還是拒絕特定請求。目標 IAM 角色必須具有正確的權限才能叫用 Lambda 函數。

1. 選擇許可索引標籤標籤。
2. 選擇新增許可，然後選擇建立內嵌政策。
3. 選擇 JSON 索引標籤，並以下列權限取代現有內容。請務必取代您的 Lambda 函數 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-2:111122223333:function:Echo"  // The
 cross-account AWS resource being accessed
    }
  ]
}
```

4. 選擇檢閱政策。
5. 在 [檢閱原則] 頁面上，輸入權限的名稱，然後選擇 [建立原則]。

步驟 4：在執行角色中添加權限以承擔目標角色

Step Functions 不會自動產生所有跨帳戶服務整合的AssumeRole政策。您必須在狀態機器的執行角色中新增所需的權限，以允許其在一或多個中擔任目標 IAM 角色 AWS 帳戶。

1. 在 IAM 主控台中開啟狀態機器的執行角色，網址為 https://console.aws.amazon.com/iam/。若要執行此作業：

   a. 在來源帳戶中開啟您在步驟 1 中建立的狀態機器。

   b. 在 [狀態機器詳細資料] 頁面上，選擇 IAM 角色 ARN。

2. 在權限索引標籤上，選擇新增權限，然後選擇建立內嵌原則。

3. 選擇 JSON 索引標籤，並以下列權限取代現有內容。請務必取代您的 Lambda 函數 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/LambdaRole"  // The target role
 to be assumed
    }
  ]
}
```

4. 選擇檢閱政策。

5. 在 [檢閱原則] 頁面上，輸入權限的名稱，然後選擇 [建立原則]。

跨帳戶存取 .sync 整合模式

當您在工作流程中使用.sync服務整合模式時，Step Functions 會輪詢叫用的跨帳號資源，以確認工作已完成。這會導致實際工作完成時間與 Step Functions 將工作識別為完成的時間之間略有延遲。目標 IAM 角色需要.sync呼叫所需的許可，才能完成此輪詢迴圈。若要這麼做，目標 IAM 角色必須具有允許來源帳戶採用的信任政策。此外，目標 IAM 角色還需要必要的許可才能完成輪詢迴圈。

> **Note**
>
> 對於巢狀 Express 工作流程，目前`arn:aws:states:::states:startExecution.sync`不支援。請改用`arn:aws:states:::aws-sdk:sfn:startSyncExecution`。

.sync 呼叫的信任原則更新

更新目標 IAM 角色的信任政策，如以下範例所示。該`sts:ExternalId`字段進一步控制誰可以擔任該角色。狀態機的名稱必須只包含 AWS Security Token Service `AssumeRole` API 支援的字元。如需詳細資訊，請參閱 AWS Security Token Service API 參考AssumeRole中的。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::sourceAccountID:role/InvokeRole",
      },
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "arn:aws:states:us-
east-2:sourceAccountID:stateMachine:stateMachineName"
        }
      }
    }
  ]
}
```

.sync 呼叫所需的權限

若要授予狀態機器所需的許可，請更新目標 IAM 角色的必要許可。如需詳細資訊，請參閱 the section called “整合式服務的 IAM 政策”。不需要範例政策中的 Amazon EventBridge 許可。例如，若要啟動狀態機器，請新增下列權限。

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

```
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:stateMachine:stateMachineName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:execution:stateMachineName:*"
      ]
    }
  ]
}
```

適用於 Step Functions 的 Amazon VPC 端點

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 託管資 AWS 源，則可以在 Amazon VPC 和 AWS Step Functions 工作流程之間建立連線。您可以將此連線與 Step Functions 工作流程搭配使用，而無需跨越公用網際網路。標準工作流程、快速工作流程和同步快速工作流程支援 Amazon VPC 端點。

Amazon VPC 可讓您在自訂虛擬網路中啟動 AWS 資源。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。如需有關 Amazon VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》。

若要將 Amazon VPC 連接到 Step Functions，您必須先定義一個介面 VPC 端點，以便將 VPC 連接到其他服務。 AWS 端點可提供可靠、可擴展的連線能力，且不需要網際網路閘道、網路地址轉譯 (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》中的界面 VPC 端點 (AWS PrivateLink)。

建立端點

您可以 AWS Step Functions 在 VPC 中使用 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS SDK、 AWS Step Functions API 或 AWS CloudFormation.

如需使用 Amazon VPC 主控台或 AWS CLI，請參閱 Amazon VPC 使用者指南中的建立界面端點。

> **Note**
>
> 建立端點時，請將 Step Functions 指定為您希望 VPC 連線到的服務。在 Amazon VPC 主控台中，服務名稱會根據 AWS 區域而有所不同。例如，如果您選擇美國東部 (維吉尼亞北部)，則「標準工作流程」和「快速工作流程」的服務名稱為 com.amazonaws.us-east-1.state，同步快速工作流程的服務名稱為 com.amazonaws.us-東部 1.sync 狀態。

> **Note**
>
> 您可以在不透過私有 DNS 覆寫 SDK 中的端點的情況下使用 VPC 端點。但是，如果您想要覆寫同步 Express 工作流程的 SDK 中的端點，則需要將`DisableHostPrefixInjection`組態設定為`true`。示例（開發套件 V2）：
>
> ```
> SfnClient.builder()
>   .endpointOverride(URI.create("https://vpce-{vpceId}.sync-states.us-
> east-1.vpce.amazonaws.com"))
>   .overrideConfiguration(ClientOverrideConfiguration.builder()
>
>  .advancedOptions(ImmutableMap.of(SdkAdvancedClientOption.DISABLE_HOST_PREFIX_INJECTION,
>  true))
>     .build())
>   .build();
> ```

如需使用建立和設定端點的相關資訊 AWS CloudFormation，請參閱使用者指南中的 AWS:: EC2:: vpcendPoint 資源。AWS CloudFormation

Amazon VPC 端點政策

若要控制對 Step Functions 的連線存取，您可以在建立 Amazon VPC 端點時附加 AWS Identity and Access Management (IAM) 端點政策。您可以透過附加多個端點政策來建立複雜的 IAM 規則。如需詳細資訊，請參閱：

- 適用於 Step Functions 的 Amazon Virtual Private Cloud 端端點政策
- 為非管理員使用者建立精細的 IAM 許可
- 使用 VPC 端點控制對服務的存取

適用於 Step Functions 的 Amazon Virtual Private Cloud 端端點政策

您可以為 Step Functions 建立 Amazon VPC 端點政策，在其中指定下列項目：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

以下範例顯示 Amazon VPC 端點政策，該政策允許一個使用者建立狀態機器，並拒絕所有其他使用者刪除狀態機器的權限。範例政策也會授予所有 使用者執行許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "*Execution",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": "*"
        },
        {
            "Action": "states:CreateStateMachine",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": {
              "AWS": "arn:aws:iam::123456789012:user/MyUser"
            }
        },
        {
            "Action": "states:DeleteStateMachine",
            "Resource": "*",
            "Effect": "Deny",
            "Principal": "*"
        }
    ]
}
```

如需建立端點政策的詳細資訊，請參閱以下內容：

- 為非管理員使用者建立精細的 IAM 許可
- 使用 VPC 端點控制對服務的存取

整合式服務的 IAM 政策

在 AWS Step Functions 主控台中建立狀態機器時，Step Functions 會根據狀態機器定義中使用的資源產生 AWS Identity and Access Management (IAM) 政策，如下所示：

- 如果您的狀態機使用其中一個最佳化整合，它將建立具有狀態機器必要權限和角色的政策。
- 如果您的狀態機使用其中一個 AWS SDK 整合，則會建立具有部分許可的 IAM 角色。之後，您可以使用 IAM 主控台新增任何遺失的角色政策。

下列範例顯示 Step Functions 如何根據您的狀態機器定義產生 IAM 政策。範例程式碼中的項目 (例如 *[[resourceName]]*) 會以狀態機器定義中所列的靜態資源來取代。如果您有多個靜態資源，則 IAM 角色中的每個資源都會有一個項目。

動態與靜態資源

靜態資源直接定義於狀態機器的任務狀態中。當您在任務狀態中包含要直接呼叫的資源相關資訊時，Step Functions 只會為這些資源建立 IAM 角色。

動態資源是傳入您的狀態中並使用路徑存取的資源 (請參閱 路徑)。如果您要將動態資源傳遞給您的工作，Step Functions 會建立更具權限的原則，指定下列項目：`"Resource": "*"`

使用執行作業模式的 Job 的其他權限

對於使用「執行作業模式」(結尾為 .sync) 的 Job，需要額外的權限才能監視和接收來自連線服務的 API 動作的回應。相關原則所包含的權限比使用要求回應或等待回呼模式的工作更多。如需同步工作服務整合模式的資訊，請參閱。

> **Note**
>
> 您需要為支援執行 Job (.sync) 模式的服務整合提供其他權限。

Step Functions 會使用兩種方法來監視工作的狀態時，在連線的服務、輪詢和事件上執行工作。

輪詢需要 Get API Describe 動作的權限，例如`ecs:DescribeTasks`或`glue:GetJobRun`。如果您的角色遺失這些權限，則 Step Functions 可能無法判斷您的工作狀態。這是因為某些執行 Job (.sync) 服務整合不支援 EventBridge事件，而某些服務只會盡力傳送事件。

從 AWS 服務傳送至 Amazon EventBridge 的事件會使用受管規則導向至 Step Functions，且需要`events:PutTargets`、`events:PutRule`和的許可`events:DescribeRule`。如果您的角色遺失

這些權限，則可能會有一段延遲，Step Functions 才會意識到您的工作完成。如需 EventBridge 事件的詳細資訊，請參閱來自 AWS 服務的事件。

> **Note**
>
> 對於執行同時支援輪詢和事件的 Job (.sync) 工作，您的工作仍可使用事件正確完成。即使您的角色缺少輪詢所需的權限，也可能發生這種情況。在這種情況下，您可能不會立即注意到輪詢權限不正確或遺失。在極少數情況下，事件無法傳遞給 Step Functions 或無法處理，您的執行可能會卡住。若要確認您的輪詢權限設定正確，您可以透過下列方式在沒有 EventBridge 事件的環境中執行執行：
>
> - 從中刪除受管理的規則 EventBridge，該規則負責將事件轉寄至 Step Functions。您帳戶中的所有狀態機器都會共用此受管規則，因此您應該只在測試或開發帳戶中執行此動作，以避免對其他狀態機器造成任何意外影響。您可以檢查目標服務之策略範本`events:PutRule`中用於的`Resource`欄位，來識別要刪除的特定受管理規則。下次您建立或更新使用該服務整合的狀態機器時，將重新建立受管理規則。如需刪除 EventBridge 規則的詳細資訊，請參閱停用或刪除規則。
> - 使用本機 Step Functions 進行測試，不支援使用事件來完成執行 Job (.sync) 工作。若要使用本機 Step Functions，請假設狀態機器使用的 IAM 角色。您可能需要編輯「信任關係」。將`AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、和`AWS_SESSION_TOKEN`環境變數設定為假定角色的值，然後使用啟動 Step Functions 數本機`java -jar StepFunctionsLocal.jar`。最後，使 AWS CLI 用 of `--endpoint-url` 參數來建立狀態機器、開始執行，以及取得執行歷程記錄。如需詳細資訊，請參閱 在本地測試狀態機。

如果停止使用「執行作業 (.sync)」模式的 Job，「Step Functions」會盡最大努力取消工作。這需要`Cancel`、`StopTerminate`、或 `Delete` API 動作的權限，例如`batch:TerminateJob`或`eks:DeleteCluster`。如果您的角色遺失這些權限，Step Functions 將無法取消您的作業，您可能會在繼續執行時產生額外費用。如需停止 Job 的詳細資訊，請參閱執行工作。

用於建立 IAM 角色的政策範本

下列主題包括當您選擇讓 Step Functions 為您建立新角色時所使用的原則範本。

> **Note**
>
> 檢閱這些範本以瞭解 Step Functions 如何建立您的 IAM 政策，以及在使用其他 AWS 服務時如何手動建立 Step Functions 的 IAM 政策範例。如需有關 Step Functions 服務整合的詳細資訊，請參閱AWS Step Functions 搭配其他服務使用。

主題

- Amazon API Gateway 的 IAM 政策
- Amazon Athena 的 IAM 政策
- 的 IAM 政策 AWS Batch
- IAM policies for Amazon Bedrock
- 的 IAM 政策 AWS CodeBuild
- 亞馬遜動態 B 的 IAM 政策
- Amazon EC/ 的 IAM 政策AWS Fargate
- Amazon EKS 的 IAM 政策
- Amazon EMR 的 IAM 政策
- EKS 上 Amazon EMR 的 IAM 政策
- Amazon EMR Serverless 的 IAM 政策
- Amazon 的 IAM 政策 EventBridge
- 的 IAM 政策 AWS Lambda
- 的 IAM 政策 AWS Glue
- 的 IAM 政策 AWS Glue DataBrew
- Amazon 的 IAM 政策 SageMaker
- Amazon SNS 的 IAM 政策
- Amazon SQS 的 IAM 政策
- 的 IAM 政策 AWS Step Functions
- 的 IAM 政策 AWS X-Ray
- 活動或沒有任何任務

Amazon API Gateway 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

資源：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

下列程式碼範例顯示呼叫 API Gateway 的資源原則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:<region>:<account-id>:<api-id>/<stage-
name>/<HTTP-VERB>/<resource-path-specifier>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": [
                        "<SourceStateMachineArn>"
                    ]
                }
            }
        }
```

```
    ]
}
```

Amazon Athena 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

`StartQueryExecution`

靜態資源

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:stopQueryExecution",
            "athena:getQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/[[workGroup]]",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
```

```
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
```

```
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/[[workGroup]]",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
```

```
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

動態資源

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
```

```
        "Action": [
            "athena:startQueryExecution",
            "athena:stopQueryExecution",
            "athena:getQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
```

```
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:startQueryExecution",
            "athena:getDataCatalog"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*",
            "arn:aws:athena:{{region}}:{{accountId}}:datacatalog/*"
        ]
    },
        {
        "Effect": "Allow",
```

```
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "glue:CreateDatabase",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:UpdateDatabase",
            "glue:DeleteDatabase",
            "glue:CreateTable",
            "glue:UpdateTable",
            "glue:GetTable",
            "glue:GetTables",
            "glue:DeleteTable",
            "glue:BatchDeleteTable",
            "glue:BatchCreatePartition",
            "glue:CreatePartition",
            "glue:UpdatePartition",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:BatchGetPartition",
            "glue:DeletePartition",
            "glue:BatchDeletePartition"
        ],
        "Resource": [
            "arn:aws:glue:{{region}}:{{accountId}}:catalog",
            "arn:aws:glue:{{region}}:{{accountId}}:database/*",
            "arn:aws:glue:{{region}}:{{accountId}}:table/*",
            "arn:aws:glue:{{region}}:{{accountId}}:userDefinedFunction/*"
        ]
    },
```

```
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": [
            "*"
        ]
    }
]
}
```

StopQueryExecution

資源

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:stopQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryExecution

資源

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryExecution"
```

```
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryResults

資源

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryResults"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": [
            "arn:aws:s3:::*"
        ]
    }
    ]
}
```

的 IAM 政策 AWS Batch

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

因為 AWS Batch 提供資源層級存取控制的部分支援，因此您必須使用。`"Resource": "*"`

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

IAM policies for Amazon Bedrock

使用控制台建立狀態機時，Step Functions會以所需的最低權限自動為狀態機建立執行角色。這些自動產生的IAM角色對於您 AWS 區域 在其中建立狀態機器有效。

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

建議您在建立IAM原則時，不要在原則中包含萬用字元。作為安全性最佳做法，您應該盡可能縮小原則的範圍。只有在執行階段期間不知道某些輸入參數時，才應使用動態原則。

在本主題中

- IAMAmazon Bedrock整合的政策範例 Step Functions

IAMAmazon Bedrock整合的政策範例 Step Functions

以下部分根據您用於特定基礎或佈建模型的 Amazon Bedrock API 描述您需要的IAM權限。本節也包含授與完整存取權的原則範例。

請記住將##文本替換為特定於資源的信息。

- IAM使用存取特定基礎模型的原則範例 InvokeModel
- IAM使用存取特定佈建模型的原則範例 InvokeModel
- 要使用的完整存取IAM原則範例 InvokeModel
- IAM將特定基礎模型作為基礎模型存取的原則範例
- IAM將特定自訂模型作為基礎模型存取的原則範例
- 使用 CreateModelCustomizationJob .sync 的完整存取IAM原則範例
- IAM使用 CreateModelCustomizationJob .sync 存取特定基礎模型的政策範例
- IAM使用 CreateModelCustomizationJob .sync 存取自訂模型的原則範例
- 使用 CreateModelCustomizationJob .sync 的完整存取IAM原則範例

IAM使用存取特定基礎模型的原則範例 InvokeModel

以下是狀態機器的IAM原則範例，該狀態機器存取`amazon.titan-text-express-v1`使用InvokeModelAPI 動作命名的特定基礎模型。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "InvokeModel1",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-v1"
            ]
        }
    ]
}
```

IAM使用存取特定佈建模型的原則範例 InvokeModel

以下是狀態機器的IAM原則範例，該狀態機器存取c2oi931ulksx使用 InvokeModelAPI 動作命名的特定佈建模型。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

要使用的完整存取IAM原則範例 InvokeModel

以下是當您使用 InvokeModelAPI 動作時提供完整存取權的狀態機器的IAM原則範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "InvokeModel1",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/*"
            ]
        }
    ]
}
```

IAM將特定基礎模型作為基礎模型存取的原則範例

以下是狀態機器使用 CreateModelCustomizationJobAPI 動作存取名amazon.titan-text-express-v1為基礎模型的特定基礎模型的IAM原則範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-
v1",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
```

```
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAM將特定自訂模型作為基礎模型存取的原則範例

以下是狀態機器使用 CreateModelCustomizationJobAPI 動作作為基礎模型存取特定自訂模型的IAM原則範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/[[roleName]]"
            ]
        }
    ]
}
```

使用 CreateModelCustomizationJob .sync 的完整存取IAM原則範例

以下是當您使用 CreateModelCustomizationJobAPI 動作時提供完整存取權的狀態機器的IAM原則範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAM使用 CreateModelCustomizationJob .sync 存取特定基礎模型的政策範例

以下是狀態機器的IAM政策範例，以存取amazon.titan-text-express-v1使用 CreateModelCustomizationJob.sync API 動作命名的特定基礎模型。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-
v1",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAM使用 CreateModelCustomizationJob .sync 存取自訂模型的原則範例

以下是狀態機器使用 CreateModelCustomizationJob.sync API 動作存取自訂模型的IAM政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

使用 CreateModelCustomizationJob .sync 的完整存取IAM原則範例

以下是當您使用 CreateModelCustomizationJob.sync API 動作時提供完整存取權的狀態機器的IAM原則範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

的 IAM 政策 AWS CodeBuild

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

資源：

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

```
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:sa-east-1:123456789012:StepFunctionsSample-
CodeBuildExecution1111-2222-3333-wJalrXUtnFEMI-SNSTopic-bPxRfiCYEXAMPLEKEY"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "codebuild:BatchGetBuilds",
                "codebuild:BatchGetReports"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForCodeBuildStartBuildRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

StartBuild

靜態資源

Run a Job (.sync)

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:BatchGetBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildRule"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
```

```
}
```

動態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:BatchGetBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildRule"
      ]
    }
  ]
}
```

Request Response

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

```
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StartBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

StopBuild

靜態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

動態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
```

```
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

BatchDeleteBuilds

靜態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

動態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

BatchGetReports

靜態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:report-group/[[reportName]]"
      ]
    }
  ]
}
```

動態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:report-group/*"
      ]
    }
  ]
}
```

StartBuildBatch

靜態資源

Run a Job (.sync)

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildBatchRule"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
```

```
    ]
}
```

動態資源

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventForCodeBuildStartBuildBatchRule"
            ]
        }
    ]
}
```

Request Response

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

StopBuildBatch

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
```

```
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

RetryBuildBatch

靜態資源

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
```

```
            ]
        }
    ]
}
```

動態資源

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch",
                "codebuild:StopBuildBatch",
                "codebuild:BatchGetBuildBatches"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:RetryBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
```

```
    ]
}
```

DeleteBuildBatch

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

亞馬遜動態 B 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:[[region]]:[[accountId]]:table/[[tableName]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

如需適用於所有 DynamoDB API 動作的身分與存取權管理政策的詳細資訊，請參閱 Amazon DynamoDB 開發人員指南中的使用 IAM 政策。此外，如需適用於 DynamoDB 之部分 SQL 的身分與存取權管理政策的相關資訊，請參閱 Amazon DynamoDB 開發人員指南中的含有 PartiQL 的 IAM 政策。

Amazon EC/ 的 IAM 政策AWS Fargate

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

由於`TaskId`的值在提交工作之後才會知道，因此 Step Functions 會建立更具權限的`"Resource": "*"`原則。

> **Note**
>
> 您只能停止由 Step Functions 啟動的 Amazon 彈性容器服務 (Amazon ECS) 任務，不論採`"*"`用 IAM 政策。

Run a Job (.sync)

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StopTask",
```

```
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:
[[accountId]]:rule/StepFunctionsGetEventsForECSTaskRule"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:StopTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
```

```
                "arn:aws:events:[[region]]:
[[accountId]]:rule/StepFunctionsGetEventsForECSTaskRule"
            ]
        }
    ]
}
```

Request Response and Callback (.waitForTaskToken)

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
}
```

如果排定的 Amazon ECS 任務需要使用任務執行角色、任務角色或任務角色覆寫，則您必須將每個任務執行角色、任務角色或任務角色覆寫的`iam:PassRole`許可新增至呼叫實體的 E CloudWatch vents IAM 角色，在此情況下為 Step Functions。

Amazon EKS 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

`CreateCluster`

資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:CreateCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:DeleteCluster"
            ],
            "Resource": "arn:aws:eks:sa-east-1:444455556666:cluster/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::444455556666:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
```

```
        }
    ]
}
```

CreateNodeGroup

資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "eks:CreateNodegroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeNodegroup",
                "eks:DeleteNodegroup"
            ],
            "Resource": "arn:aws:eks:sa-east-1:444455556666:nodegroup/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "arn:aws:iam::444455556666:role/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::444455556666:role/StepFunctionsSample-EKSClusterMan-
NodeInstanceRole-ANPAJ2UCCR6DPCEXAMPLE"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

DeleteCluster

資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteCluster",
                "eks:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:cluster/ExampleCluster"
            ]
        }
    ]
}
```

DeleteNodegroup

資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteNodegroup",
                "eks:DescribeNodegroup"
            ],
```

```
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:nodegroup/ExampleCluster/
ExampleNodegroup/*"
            ]
        }
    ]
}
```

如需將 Amazon EKS 與 Step Functions 搭配使用的詳細資訊，請參閱使用 Step Functions 調用 Amazon EKS。

Amazon EMR 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

`addStep`

靜態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

動態資源

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

```
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

cancelStep

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

createCluster

靜態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:RunJobFlow",
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:TerminateJobFlows"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": [
        "arn:aws:iam::{{account}}:role/[[roleName]]"
      ]
    }
  ]
}
```

setClusterTerminationProtection

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

modifyInstanceFleetByName

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

modifyInstanceGroupByName

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

terminateCluster

靜態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

動態資源

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

EKS 上 Amazon EMR 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

CreateVirtualCluster

資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:CreateVirtualCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{{accountId}}:role/aws-service-role/emr-
containers.amazonaws.com/AnAWSServiceRoleForAmazonEMRContainers",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "emr-containers.amazonaws.com"
                }
            }
        }
    ]
}
```

DeleteVirtualCluster

靜態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster",
        "emr-containers:DescribeVirtualCluster"
      ],
```

```
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ]
    }
  ]
}
```

動態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster",
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ]
```

```
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DeleteVirtualCluster"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ]
    }
  ]
}
```

`StartJobRun`

靜態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
```

```
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun",
        "emr-containers:CancelJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]/jobruns/*"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/
[[virtualClusterId]]"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    }
  ]
}
```

動態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun",
        "emr-containers:CancelJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ]
    }
  ]
}
```

Request Response

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": [
        "arn:aws:emr-containers:{{region}}:{{accountId}}:/virtualclusters/*"
      ],
```

```
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "[[executionRoleArn]]"
          ]
        }
      }
    }
  ]
}
```

Amazon EMR Serverless 的 IAM 政策

使用控制台建立狀態機時，Step Functions會以所需的最低權限自動為狀態機建立執行角色。這些自動產生的IAM角色對於您 AWS 區域 在其中建立狀態機器有效。

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

建議您在建立IAM原則時，不要在原則中包含萬用字元。作為安全性最佳做法，您應該盡可能縮小原則的範圍。只有在執行階段期間不知道某些輸入參數時，才應使用動態原則。

此外，管理員使用者在授與執行狀態機器的非系統管理員使用者執行角色時，應該小心。如果您要自行建立原則，建議您在執行角色中包含 passRole 原則。我們也建議您在執行角色中新增`aws:SourceARN`和`aws:SourceAccount`內容索引鍵。

在本主題中

- EMR 無伺服器與 Step Functions 整合的 IAM 政策範例

EMR 無伺服器與 Step Functions 整合的 IAM 政策範例

- IAM 政策範例 CreateApplication
- IAM 政策範例 StartApplication
- IAM 政策範例 StopApplication
- IAM 政策範例 DeleteApplication
- IAM 政策範例 StartJobRun
- IAM 政策範例 CancelJobRun

IAM 政策範例 CreateApplication

以下是具有狀態之狀態機器的 IAM 政策範例。 CreateApplication 任務

> **Note**
>
> 您必須在帳戶中建立第一個應用程式期間，在 IAM 政策中指定 CreateServiceLinkedRole 許可。此後，您無需添加此權限。如需有關的資訊 CreateServiceLinkedRole，請參閱 https://docs.aws.amazon.com/IAM/latest/APIReference/ CreateServiceLinkedRole中的。

下列原則的靜態和動態資源相同。

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:GetApplication",
                "emr-serverless:DeleteApplication"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
```

```
            ],
            "Resource": [
                "arn:aws:events:{{region}}:{{accountId}}:rule/
StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{{accountId}}:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWS ServiceRoleForAmazonEMRServerless*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

Request Response

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

   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::{{accountId}}:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWS ServiceRoleForAmazonEMRServerless*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"
```

```
                }
            }
        }
    ]
}
```

IAM 政策範例 StartApplication

靜態資源

以下是使用具有狀態的狀態機器時，靜態資源的 IAM 政策範例。 StartApplication 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication",
               "emr-serverless:GetApplication",
               "emr-serverless:StopApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
```

```
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        }
    ]
}
```

動態資源

以下是使用具有狀態的狀態機器時，動態資源的 IAM 政策範例。 StartApplication 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication",
               "emr-serverless:GetApplication",
               "emr-serverless:StopApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        }
    ]
}
```

IAM 政策範例 StopApplication

靜態資源

以下是使用具有狀態的狀態機器時，靜態資源的 IAM 政策範例。 StopApplication 任務

Run a Job (.sync)

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

```
            "Effect": "Allow",
            "Action": [
                "emr-serverless:StopApplication",
                "emr-serverless:GetApplication"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
             ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
             ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:StopApplication"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
             ]
        }
    ]
}
```

動態資源

以下是使用具有狀態的狀態機器時，動態資源的 IAM 政策範例。 StopApplication 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StopApplication",
               "emr-serverless:GetApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StopApplication"
```

```
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        }
    ]
}
```

IAM 政策範例 DeleteApplication

靜態資源

以下是使用具有狀態的狀態機器時，靜態資源的 IAM 政策範例。 DeleteApplication 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication",
               "emr-serverless:GetApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
```

```
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        }
    ]
}
```

動態資源

以下是使用具有狀態的狀態機器時，動態資源的 IAM 政策範例。 DeleteApplication 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication",
               "emr-serverless:GetApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessApplicationRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:DeleteApplication"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        }
    ]
}
```

IAM 政策範例 StartJobRun

靜態資源

以下是使用具有狀態的狀態機器時，靜態資源的 IAM 政策範例。 StartJobRun 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
```

```
            "Action": [
                "emr-serverless:StartJobRun"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
             ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "[[jobExecutionRoleArn]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:GetJobRun",
                "emr-serverless:CancelJobRun"
            ],
            "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]/jobruns/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
```

```
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:StartJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "[[jobExecutionRoleArn]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

動態資源

以下是使用具有狀態的狀態機器時，動態資源的 IAM 政策範例。 StartJobRun 任務

Run a Job (.sync)

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

```
            "Effect": "Allow",
            "Action": [
                "emr-serverless:StartJobRun",
                "emr-serverless:GetJobRun",
                "emr-serverless:CancelJobRun"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
             ]
        },
        {
             "Effect": "Allow",
             "Action": "iam:PassRole",
             "Resource": [
                 "[[jobExecutionRoleArn]]"
             ],
             "Condition": {
                 "StringEquals": {
                     "iam:PassedToService": "emr-serverless.amazonaws.com"
                 }
             }
        },
        {
             "Effect": "Allow",
             "Action": [
                 "events:PutTargets",
                 "events:PutRule",
                 "events:DescribeRule"
             ],
             "Resource": [
                 "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
             ]
        }
    ]
}
```

Request Response

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

```
            "Effect": "Allow",
            "Action": [
                "emr-serverless:StartJobRun"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
             ]
        },
        {
             "Effect": "Allow",
             "Action": "iam:PassRole",
             "Resource": [
                 "[[jobExecutionRoleArn]]"
             ],
             "Condition": {
                 "StringEquals": {
                     "iam:PassedToService": "emr-serverless.amazonaws.com"
                 }
             }
        }
    ]
}
```

IAM 政策範例 CancelJobRun

靜態資源

以下是使用具有狀態的狀態機器時，靜態資源的 IAM 政策範例。 CancelJobRun 任務

Run a Job (.sync)

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CancelJobRun",
                "emr-serverless:GetJobRun"
             ],
            "Resource": [
                 "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]/jobruns/[[jobRunId]]"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:CancelJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/
[[applicationId]]/jobruns/[[jobRunId]]"
            ]
        }
    ]
}
```

動態資源

以下是使用具有狀態的狀態機器時，動態資源的 IAM 政策範例。 CancelJobRun 任務

Run a Job (.sync)

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:CancelJobRun",
               "emr-serverless:GetJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:{{region}}:
{{accountId}}:rule/StepFunctionsGetEventsForEMRServerlessJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
           "Effect": "Allow",
           "Action": [
               "emr-serverless:CancelJobRun"
            ],
           "Resource": [
                "arn:aws:emr-serverless:{{region}}:{{accountId}}:/applications/*"
            ]
        }
    ]
}
```

Amazon 的 IAM 政策 EventBridge

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

`PutEvents`

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:123456789012:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

動態資源

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

如需 EventBridge 與 Step Functions 搭配使用的更多資訊，請參閱 EventBridge 使用 Step Functions 調用。

的 IAM 政策 AWS Lambda

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

AWS Step Functions 根據您的狀態機器定義產生 IAM 政策。對於具有兩個呼叫`function1`和的 AWS Lambda 工作狀態的狀態機器`function2`，必須使用具有這兩個函數`lambda:Invoke`權限的原則。

如以下範例所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:[[region]]:[[accountId]]:function:[[function1]]",
                "arn:aws:lambda:[[region]]:[[accountId]]:function:[[function2]]"
            ]
        }
    ]
}
```

的 IAM 政策 AWS Glue

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

AWS Glue 沒有基於資源的控制。

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "glue:StartJobRun",
                "glue:GetJobRun",
                "glue:GetJobRuns",
                "glue:BatchStopJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

的 IAM 政策 AWS Glue DataBrew

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

Run a Job (.sync)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:startJobRun",
                "databrew:listJobRuns",
                "databrew:stopJobRun"
```

```
            ],
            "Resource": [
                "arn:aws:databrew:{{region}}:{{accountId}}:job/*"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "databrew:startJobRun"
            ],
            "Resource": [
                "arn:aws:databrew:{{region}}:{{accountId}}:job/*"
            ]
        }
    ]
}
```

Amazon 的 IAM 政策 SageMaker

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

> **Note**
>
> 在這些範例中，請*[[roleArn]]*參閱 IAM 角色的 Amazon 資源名稱 (ARN)，該角色用於存取 SageMaker 用於在 ML 運算執行個體上部署的模型成品和 docker 映像，或用於批次轉換任務。如需詳細資訊，請參閱 Amazon SageMaker 角色。

`CreateTrainingJob`

靜態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:StopTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-
job/[[trainingJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
```

```
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-
job/[[trainingJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

動態資源

.sync or .waitForTaskToken

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob",
        "sagemaker:DescribeTrainingJob",
        "sagemaker:StopTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
```

```
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTrainingJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:training-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
```

```
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

`CreateTransformJob`

> **Note**
>
> AWS Step Functions 當您建立與之整合的狀態機器`CreateTransformJob`時，不會自動建立原則 SageMaker。您必須根據下列其中一個 IAM 範例，將內嵌政策附加至建立的角色。

靜態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob",
        "sagemaker:DescribeTransformJob",
```

```
        "sagemaker:StopTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-
job/[[transformJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-
job/[[transformJobName]]*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

動態資源

Run a Job (.sync)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob",
        "sagemaker:DescribeTransformJob",
        "sagemaker:StopTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "events:PutTargets",
        "events:PutRule",
        "events:DescribeRule"
      ],
      "Resource": [
        "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule"
      ]
    }
  ]
}
```

Request Response and Callback (.waitForTaskToken)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:CreateTransformJob"
      ],
      "Resource": [
        "arn:aws:sagemaker:[[region]]:[[accountId]]:transform-job/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListTags"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "[[roleArn]]"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "sagemaker.amazonaws.com"
        }
      }
    }
  ]
}
```

Amazon SNS 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
            ]
        }
    ]
}
```

以路徑為基礎的資源，或發佈至*TargetArn*或 *PhoneNumber*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon SQS 的 IAM 政策

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

靜態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
}
```

動態資源

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

的 IAM 政策 AWS Step Functions

對於要求單一巢狀工作流程執行的狀態機器，請使用將權限限制`StartExecution`為該狀態機器的 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

如需詳細資訊，請參閱下列內容：

- AWS Step Functions 搭配其他服務使用
- 將參數傳遞至服務 API
- 以整合式服務 AWS Step Functions 的形式管理執行

Synchronous

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
```

```
            ],
            "Resource": [
                "arn:aws:states:[[region]]:[[accountId]]:stateMachine:
[[stateMachineName]]"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "states:DescribeExecution",
                "states:StopExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:execution:[[stateMachineName]]:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForStepFunctionsExecutionRule"
            ]
        }
    ]
}
```

Asynchronous

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [
```

```
            "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
                ]
            }
        ]
    }
```

如需巢狀工作流程執行的詳細資訊，請參閱從任務狀態開始工作流程執行。

的 IAM 政策 AWS X-Ray

下列範例範本顯示如何 AWS Step Functions 根據狀態機器定義中的資源產生 IAM 政策。如需詳細資訊，請參閱 整合式服務的 IAM 政策 及 服務整合模式。

若要啟用 X-Ray 追蹤，您需要具有適當權限的 IAM 政策以允許追蹤。如果您的狀態機使用其他整合式服務，則可能需要其他 IAM 政策。請參閱特定服務整合的 IAM 政策。

當您建立啟用 X-Ray 追蹤的狀態機時，會自動建立 IAM 政策。

> **Note**
>
> 如果您為現有狀態機器啟用 X-Ray 追蹤，您必須確保新增具有足夠權限的原則才能啟用 X-Ray 追蹤。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
            "Resource": [
                "*"
            ]
        }
```

```
    ]
}
```

如需將 X-Ray 與 Step Functions 搭配使用的更多資訊，請參閱AWS X-Ray 和 Step Functions。

活動或沒有任何任務

對於只有任務或完全沒有`Activity`任務的狀態機器，請使用拒絕存取所有動作和資源的 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

如需使用 `Activity` 任務的詳細資訊，請參閱活動。

使用分散式地圖狀態的 IAM 政策

使用 Step Functions 主控台建立工作流程時，Step Functions 可以根據工作流程定義中的資源自動產生 IAM 政策。這些原則包括允許狀態機器角色呼叫分散式對應狀態的 `StartExecution` API 動作所需的最低權限。這些政策還包括存取 AWS 資源所需的最低權限 Step Functions，例如 Amazon S3 儲存貯體和物件以及 Lambda 函數。我們強烈建議您僅在 IAM 政策中加入必要的許可。例如，如果您的工作流程包含分散式模式的`Map`狀態，請將您的政策範圍縮減到包含資料集的特定 Amazon S3 儲存貯體和資料夾。

> **Important**
>
> 如果您指定 Amazon S3 儲存貯體和物件或前置詞，其中包含分散式地圖狀態輸入中現有鍵值組的參考路徑，請確定您已更新工作流程的 IAM 政策。將政策範圍縮小到值區和路徑在執行時期解析為的物件名稱。

在本主題中：

- 執行分散式地圖狀態的 IAM 政策範例
- 分散式地圖的 IAM 政策範例 redriving
- 從 Amazon S3 資料集讀取資料的 IAM 政策範例
- 將資料寫入 Amazon S3 儲存貯體的 IAM 政策範例

執行分散式地圖狀態的 IAM 政策範例

當您在工作流程中包含分散式對應狀態時，Step Functions 需要適當的權限，才能允許狀態機器角色呼叫分散式對應狀態的 `StartExecution` API 動作。

以下 IAM 政策示例授予狀態機器角色執行分散式地圖狀態所需的最少權限。

> **Note**
>
> 請確定您以使*stateMachineName*用分散式地圖狀態的狀態機器名稱取代。例如 arn:aws:states:*us-east-2*:*123456789012*:stateMachine:*mystateMachine*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "arn:aws:states:region:accountID:stateMachine:stateMachineName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:region:accountID:execution:stateMachineName:*"
    }
  ]
}
```

分散式地圖的 IAM 政策範例 redriving

您可以在父工作流程的 Map Run 中重新啟動不成功redriving的子工作流程執行。redriven父工作流程 redrives所有不成功的狀態，包括分佈式地圖。請確定您的執行角色具有允許其在父工作流程上叫用 RedriveExecution API 動作所需的最低權限。

以下 IAM 政策示例授予分散式 Map 狀態中狀態機器角色所需redriving的最少權限。

> **Note**
>
> 請確定您以使*stateMachineName*用分散式地圖狀態的狀態機器名稱取代。例如 arn:aws:states:*us-east-2*:*123456789012*:stateMachine:*mystateMachine*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

從 Amazon S3 資料集讀取資料的 IAM 政策範例

下列 IAM 政策範例授予使用 ListObjectsV2 和 GetObjectAPI 動作存取 Amazon S3 資料集所需的最低權限。

Example 適用於作為資料集的 Amazon S3 物件的 IAM 政策

下列範例顯示一項 IAM 政策，該政策授與存取名為的 Amazon S3 儲存貯體*processImages*中組織的物件的最少權限*myBucket*。

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

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example 將 CSV 檔案作為資料集的 IAM 政策

下列範例顯示授與存取名為 CSV 檔案的最少權限的 IAM 政策*ratings.csv*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example 將 Amazon S3 庫存作為資料集的 IAM 政策

下列範例顯示授與存取 Amazon S3 庫存報告的最少權限的 IAM 政策。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-
MMZ/manifest.json",
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/data/*"
            ]
        }
    ]
}
```

將資料寫入 Amazon S3 儲存貯體的 IAM 政策範例

下列 IAM 政策範例授予使用 PutObject API 動作將子工作流程執行結果寫入 Amazon S3 儲存貯體中名為 *CSVJobs* 的資料夾所需的最低權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

AWS KMS key 加密 Amazon S3 儲存貯體的 IAM 許可

分散式地圖狀態使用分段上傳將子工作流程執行結果寫入 Amazon S3 儲存貯體。如果儲存貯體是使用AWS Key Management Service(AWS KMS)金鑰加密的，您還必須在IAM原則中包含權限

kms:Decryptkms:Encrypt，才能對金鑰執行、和kms:GenerateDataKey動作。這些許可是必要的，因為在加密檔案完成分段上傳之前，Amazon S3 必須從部分加密檔案解密並讀取資料。

以下 IAM 政策範例授予對用於kms:Decrypt加密 Amazon S3 儲存貯體之金鑰的kms:Encrypt、和kms:GenerateDataKey動作的許可。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:Encrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": [
      "arn:aws:kms:us-east-1:123456789012:key/111aa2bb-333c-4d44-5555-a111bb2c33dd"
    ]
  }
}
```

如需詳細資訊，請參閱 AWS 知識中心中的使用 AWS KMS key透過加密以將大型檔案上傳至 Amazon S3。

如果您的 IAM 使用者或角色與相 AWS 帳戶 同KMS key，則您必須擁有金鑰政策的這些許可。如果您的 IAM 使用者或角色屬於與帳戶不同的帳戶KMS key，則您必須同時擁有金鑰政策和 IAM 使用者或角色的許可。

標籤類型政策

Step Functions 支援以標記為基礎的原則。例如，您可以限制所有 Step Functions 資源的存取權，這些資源包含含索引鍵environment和值的標籤production。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:UntagResource",
                "states:DeleteActivity",
```

```
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

此政策會 Deny 提供針對已標記為 environment/production 所有資源刪除狀態機器或活動、停止執行，以及新增或刪除新標籤的能力。

對於以標籤為基礎的授權，如下列範例所示的狀態機器執行資源會繼承與狀態機器相關聯的標籤。

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

當您呼叫DescribeExecution或其他指定執行資源 ARN 的 API 時，Step Functions 會在執行以標籤為基礎的授權時，使用與狀態機器相關聯的標籤來接受或拒絕要求。這可協助您在狀態機器層級允許或拒絕存取狀態機器執行。

如需標記的詳細資訊，請參閱以下內容：

- 步驟函數中的標記
- 使用 IAM 標籤控制存取

疑難排解 AWS Step Functions 身分和存取

使用下列資訊可協助您診斷和修正使用 Step Functions 和 IAM 時可能會遇到的常見問題。

主題

- 我沒有授權在 Step Functions 中執行動作
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪問我 AWS 帳戶 的 Step Functions 資源

我沒有授權在 Step Functions 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在 mateojackson 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 states:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 Mateo 政策，允許他使用 states:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的原則，以允許您將角色傳遞給 Step Functions。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台在 Step Functions 中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪問我 AWS 帳戶 的 Step Functions 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解 Step Functions 是否支援這些功能，請參閱如何與 IAM AWS Step Functions 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。

- 若要了解如何向第三方提供對資源的存取權 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策的差異。

記錄和監控

如需中記錄和監控的相關資訊 AWS Step Functions，請參閱〈〉一記錄和監控節。

符合性驗證 AWS Step Functions

協力廠商稽核員會評估其安全性與合規性， AWS Step Functions 做為多個 AWS 合規計畫的一部分。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定規範計劃範圍內的 AWS 服務清單，請參閱合規計劃AWS 服務範圍方案)。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

使用 Step Functions 時，您的合規責任取決於資料的敏感度、公司的合規目標以及適用的法律和法規。 AWS 提供下列資源以協助遵循法規：

- 安全性與合規快速入門指南 — 這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。 AWS
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 標準的應 AWS 用程式。
- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

韌性在 AWS Step Functions

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎架構之外，Step Functions 還提供多種功能，以協助支援您的資料恢復能力和備份需求。

基礎架構安全 AWS Step Functions

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

您可以使用 AWS 已發佈的 API 呼叫透過網路存取 Step Functions。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可以從任何網路位置呼叫 AWS API 作業，但Step Functions不支援以資源為基礎的存取原則，其中可能包含以來源 IP 位址為基礎的限制。您也可以使用 Step Functions 政策，從特定 Amazon Virtual Private Cloud (Amazon VPC) 端點或特定 VPC 控制存取。實際上，這會將對特定Step Functions資源的網路存取從網路內的特定 VPC 隔離出來 AWS 。

中的組態和弱點分析 AWS Step Functions

配置和 IT 控制是與您（我們的客戶） AWS 之間的共同責任。如需詳細資訊，請參閱 AWS 共用的責任模型。

將工作負載從移轉 AWS Data Pipeline 至 Step Functions

AWS 於二零一二年推出該 AWS Data Pipeline 服務。當時，客戶希望能夠使用各種運算選項在不同資料來源之間移動資料的服務。隨著數據傳輸需求隨著時間的推移而改變，因此可以滿足這些需求。您現在可以選擇最符合您業務需求的解決方案。例如，您可以執行下列任何操作：

- 使用 Step Functions 來協調多個 AWS 服務之間的工作流程。
- 使用 Amazon 管理的 Apache 氣流工作流程 (Amazon MWAA) 來管理 Apache 氣流的工作流程協調流程。
- 用 AWS Glue 於執行和協調 Apache 星火應用程式。

您可以將的典型使用案例遷移 AWS Data Pipeline 到 Step Functions 或 Amazon MWAA。 AWS Glue您選擇的選項取決於您目前的工作負載 AWS Data Pipeline。本主題說明如何從移轉 AWS Data Pipeline 至 Step Functions。

主題

從移轉工作負載 AWS Data Pipeline

Step Functions 是無伺服器協調服務，您可以在其中為關鍵業務應用程式建置工作流程。透過 Step 函式的工作流程 Studio，您可以建置工作流程，並將它們與超過 250 個 11,000 個 API 動作整合。 AWS 服務這包括 AWS 服務 如 AWS Lambda Amazon EMR 和 Amazon DynamoDB。您也可以使用 Step Functions 來協調資料處理管線、處理錯誤，以及處理基礎上的節流限制。 AWS 服務您可以建立工作流程，以處理和發佈機器學習模型、協調微服務，以及使用來處理擷取、轉換和載入 (ETL) 工作流程。 AWS Glue您也可以為需要人工互動的應用程式建立長時間執行的自動化工作流程。

Step Functions 是由提供的全受管服務 AWS。這表示您可以AWS 管理諸如維護基礎結構、修補 Worker 以及管理作業系統版本更新等工作。

當您的使用案例符合下列條件時，建議您從「步驟函數」移轉 AWS Data Pipeline 至「Step Functions」：

- 您偏好無伺服器、高可用性的工作流程協調服務。
- 您需要以單一工作執行的精細程度收費的解決方案。
- 您的工作負載涉及為多個其他工作負載協調任務 AWS 服務，例如 Amazon EMR AWS Glue、Lambda 或 DynamoDB。
- 您需要具有 drag-and-drop 視覺化設計工作流程建立的低程式碼解決方案。這個解決方案不需要學習陌生、複雜的程式設計概念。
- 您需要整合超過 250 個涵蓋 11,000 AWS 服務 個 API 動作的服務。此服務還必須與以外的自訂服務和活動整合 AWS。

Step Functions 和之間的概念映射 AWS Data Pipeline

AWS Data Pipeline 和 Step Functions 共用一些共同的概念。例如，若要定義工作流程，您可以在 AWS Data Pipeline 和 Step Functions 中使用 JSON 格式。在 Step Functions 中，您可以使用Amazon States Language，這是一種基於 JSON 的結構化語言。您可以使用 Amazon 州語言 (ASL) 定義工作流程，並在工作流程的文字和視覺表示之間切換。這種 JSON 格式有助於簡化將工作流程儲存在原始檔控制工具中。它還可以幫助您管理多個版本的工作流程，控制其訪問權限，或使用 CI/CD 方法自動化其協調。

下表說明這兩個服務中使用的主要概念之間的對應。左側的「資料管線概念」欄會列出中的概念 AWS Data Pipeline，而右邊的「Step Functions 數」概念欄則列出「Step Functions 數」中的對等概念。

資料管線概念	Step Functions 概念
管道	工作流
管道定義	Amazon States Language(ASL)
活動	狀態 和 任務
執行個體	執行
Attempts	捕手和取回器
管線排程	• 使用 Amazon EventBridge 調度程序執行 • 透過 EventBridge 管道觸發的事件
管線運算式和函數	• 內部函數

資料管線概念	Step Functions 概念
	• 使用服務整合的 Lambda 函

Step Functions 範例專案

如需 Step Functions 的簡介，請參閱下列視訊：

開始使用 AWS Step Functions 服務協調

下列清單概述了一些使用 Step Functions 實作最常見 AWS Data Pipeline 使用案例的範例專案。您可以使用這些範例專案做為從「步驟函數」移轉 AWS Data Pipeline 到「Step Functions」的參考。您也可以將它們用作樣板，以建立自己的工作流程，並 AWS 服務根據您的使用案例與支援的工作流程整合。

- 管理 Amazon EMR Job
- 在 Amazon EMR 無伺服器上執行資料處理任務
- 運行/豬/Hadoop 工作
- 查詢大型資料集 (Amazon Athena、Amazon S3 AWS Glue、Amazon SNS)
- 使用 Amazon Redshift 執行 ETL/ELT 工作流程
- 編排爬蟲 AWS Glue
- 使用 Step Functions 執行殼層指令碼

若要深入了解 Step Functions，請參閱下列主題與資源：

- Step Functions 的教學課程
- Step Functions 的範例專案
- AWS Step Functions 工作坊

價格比較

AWS Data Pipeline 按管道數量及其使用級別定價。每天執行一次以上 (高頻率) 的活動定價為每月每個活動 $1。每天執行一次或更少 (低頻率) 的活動定價為每月每個活動 $0.60。非作用中管道的價格為每個管線 1 美元。如需有關定價的詳細資訊，請參閱AWS Data Pipeline 定價頁面。

Step Functions 有兩種類型的工作流程：標準和快速。每個工作流程類型都有不同的定價模式。此比較是以「標準」工作流程為基礎，因為它最符合的常見使用案例 AWS Data Pipeline。標準工作流程的價格為每 1000 個狀態轉換 0.025 美元。非作用中的狀態機器不需要任何費用；您只需按使用量付費。如需有關定價的詳細資訊，請參閱AWS Step Functions 定價頁面。

疑難排解

如果您在使用步驟函數時遇到問題，請使用下列疑難排解資源。

主題

一般性問題的故障診斷

我無法創建狀態機。

與狀態機器關聯的 IAM 角色可能沒有足夠的權限。檢查 IAM 角色的許可，包括AWS服務整合任務、X-Ray 和CloudWatch記錄。.sync任務狀態需要其他權限。

我無法使用 a JsonPath 來引用前一個任務的輸出。

對於JsonPath，JSON 金鑰必須以結尾.$。這意味著 a 只JsonPath能在鍵值對中使用。如果你想使用JsonPath其他地方，如一個數組，你可以使用內在函數。例如，您可以使用類似下列的項目：

任務 A 輸出：

```
{
    "sample": "test"
}
```

任務 B：

```
{
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> 使用 Step Functions 主控台中的資料流程模擬器來測試 JSON 路徑語法、更好地瞭解如何在狀態內操作資料，並查看資料在狀態之間傳遞的方式。

狀態轉換有延遲。

對於標準工作流程，狀態轉換的數目有限制。當您超過狀態轉換限制時，Step Functions 會延遲狀態轉換，直到滿足配額的值區為止。您可以複查 執行指標「`ExecutionThrottled`測CloudWatch量結果」頁面段落中的測量結果，來監督狀態轉換限制節流。

當我開始新的標準工作流程執行時，它們會失敗並顯示**`ExecutionLimitExceeded`**錯誤。

步驟函數對每個函數AWS 帳戶中的每個執行都有 1,000,000 個開啟的執行的限制。AWS 區域如果超過此限制，步驟函數會擲回錯`ExecutionLimitExceeded`誤。此限制不適用於快速工作流程。您可以使用 Amazon 使用CloudWatch者指南中的下列指CloudWatch標數學來估算開啟的執行次數：`ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`

處於並行狀態的一個分支上的失敗會導致整個執行失敗。

這是預期的行為。若要避免在使用平行狀態時發生失敗，請設定 Step Functions 以捕捉從每個分支擲回的錯誤。

疑難排解服務整

我的工作已在下游服務中完成，但在「步驟函數」中，任務狀態仍為「進行中」或其完成延遲。

對於`.sync`服務整合模式，Step Functions 會使用EventBridge規則、下游 API 或兩者的組合來偵測下游工作狀態。對於某些服務，步驟函數不會建立要監視的EventBridge規則。例如，對於AWS Glue服務整合，而不是使用EventBridge規則，步驟函數會進行`glue:GetJobRun`呼叫。由於 API 呼叫的頻率，下游工作完成與「步驟函式」工作完成時間之間存在差異。Step Functions 需要 IAM 許可才能管理EventBridge規則並呼叫下游服務。如需執行角色權限不足會如何影響工作完成的詳細資訊，請參閱使用執行作業模式的 Job 的其他權限。

我想從嵌套狀態機執行返回 JSON 輸出。

步驟函數有兩種步驟函數同步服務整合：`startExecution.sync`和`startExecution.sync:2`。兩者都等待嵌套狀態機完成，但它們返回不同的`Output`格式。您可以使`startExecution.sync:2`用在下返回 JSON 輸出`Output`。

我無法從另一個帳戶叫用 Lambda 函數。

透過跨帳戶支援存取 Lambda 函數

如果您的區域中可以跨帳戶存取AWS資源，請使用下列方法從另一個帳戶叫用 Lambda 函數。

若要在工作流程中叫用跨帳號資源，請執行下列動作：

1. 在包含資源的目標帳戶中建立 IAM 角色。此角色會授與包含狀態機器的來源帳戶存取目標帳戶資源的權限。
2. 在`Task`州的定義中，指定狀態機器在叫用跨帳戶資源之前要採用的目標 IAM 角色。
3. 修改目標 IAM 角色中的信任政策，以允許來源帳戶暫時擔任此角色。信任政策必須包含來源帳戶中定義之狀態機器的 Amazon 資源名稱 (ARN)。此外，請在目標 IAM 角色中定義適當的許可，以呼叫AWS資源。
4. 更新來源帳戶的執行角色，以包含假設目標 IAM 角色所需的權限。

如需範例，請參閱 教學課程：存取跨帳戶 AWS 資源。

> **Note**
>
> 您可以將狀態機設定為擔任 IAM 角色，以便從多個存取資源AWS 帳戶。但是，狀態機器在給定時間只能承擔一個 IAM 角色。

如需指定跨帳號資源的`Task`狀態定義範例，請參閱作業狀態的證明資料欄位範例。

存取 Lambda 函數，而不需要跨帳戶支援

如果您的區域無法跨帳戶存取AWS資源，請使用下列方法從其他帳戶叫用 Lambda 函數。

在`Task`州/省的`Resource`字段中，使用`arn:aws:states:::lambda:invoke`並傳遞`FunctionArn` in 參數。與狀態機器相關聯的 IAM 角色必須具有正確的許可才能叫用跨帳戶 Lambda 函數:`lambda:invokeFunction`。

```
{
   "StartAt":"CallLambda",
   "States":{
      "CallLambda":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-west-2:123456789012:function:my-function"
         },
         "End":true
      }
   }
}
```

我無法看到從.waitForTaskToken州傳遞的任務令牌。

在Task州/省的Parameters字段中，您必須傳遞任務令牌。例如，您可以使用類似下列程式碼的項目。

```
{
   "StartAt":"taskToken",
   "States":{
      "taskToken":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke.waitForTaskToken",
         "Parameters":{
            "FunctionName":"get-model-review-decision",
            "Payload":{
               "token.$":"$$.Task.Token"
            },
         },
         "End":true
      }
   }
}
```

> **Note**
>
> 您可以嘗試使.waitForTaskToken用任何 API 動作。但是，某些 API 沒有任何合適的參數。

疑難排解活

我的狀態機執行卡在活動狀態。

在您使用 GetActivityTaskAPI 動作輪詢任務 Token 之前，活動任務狀態不會啟動。最佳作法是新增工作層級逾時，以避免卡住的執行。如需詳細資訊，請參閱使用超時來避免卡住的執行。

如果您的狀態機卡在ActivityScheduled事件中，則表示您的活動工作者叢集有問題或規模不足。您應該監視指ActivityScheduleTimeCloudWatch標並在該時間增加時設置警報。但是，若要逾時狀態未轉換為狀態的任何卡住Activity狀態機器執行，請在ActivityStarted狀態機器層級定義逾時。若要執行此操作，請在狀態機定義的開頭指定TimeoutSeconds欄位，在States欄位之外。

我的活動工作者在等待任務令牌時超時。

Worker 使用 GetActivityTaskAPI 動作來擷取具有指定活動 ARN 的任務，該作業已排定由執行中狀態機器執行。 GetActivityTask啟動長輪詢，所以服務保持 HTTP 連接打開，並在任務變為可用時立即響應。服務在回應之前保留要求的時間上限為 60 秒。如果 60 秒內沒有任務可用，則輪詢返回一taskToken個空字符串。若要避免此逾時，請在 AWS SDK 或您用來進行 API 呼叫的用戶端用戶端通訊端設定逾時至少 65 秒的逾時時間。

疑難排解快速工

我的應用程序在從 StartSyncExecution API 調用收到響應之前超時。

在您用來進行 API 呼叫的 AWS SDK 或用戶端中設定用戶端通訊端逾時。若要接收回應，逾時值必須高於「快速工作流程」執行的持續時間。

我無法看到執行歷程記錄，以便疑難排解 Express 工作流程失敗。

Express 工作流程不會在中記錄執行歷程記錄AWS Step Functions。相反地，您必須開啟CloudWatch記錄功能。開啟記錄功能後，您可以使用CloudWatch日誌深入解析查詢來檢閱 Express 工作流程執行。如果您選擇「執行」標籤中的「啟用」按鈕，也可以在「步驟函數」主控台上檢視 Express Workflow 執行的執行歷史記錄。如需詳細資訊，請參閱在 Step Functions 主控台上檢視和偵錯執行。

要根據持續時間列出執行：

```
fields ispresent(execution_arn) as exec_arn
| filter exec_arn
```

```
| filter type in ["ExecutionStarted", "ExecutionSucceeded", "ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
| stats latest(type) as status,
  tomillis(earliest(event_timestamp)) as UTC_starttime,
  tomillis(latest(event_timestamp)) as UTC_endtime,
  latest(event_timestamp) - earliest(event_timestamp) as duration_in_ms  by
 execution_arn
| sort duration desc
```

若要列出失敗和已取消的執行項目：

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

相關資訊

下表列出在您使用此服務時可能有用的相關資源。

資源	描述
AWS Step Functions API 參考	API 動作、參數和資料類型的說明，以及服務會傳回的錯誤清單。
AWS Step Functions指令行參考	可與 AWS Step Functions 搭配使用的 AWS CLI 命令說明。
步進函數的產品資訊	步驟函數相關資訊的主要網頁。
開發論壇	以社群為基礎的論壇，供開發人員討論與 Step Functions 和其他AWS服務相關的技術問題。
AWS Support信息	有關資訊的主要網頁, 或AWS Support, 快速回應支援通道ne-on-one，以說明您建置和執行AWS基礎結構服務上的應用程式.

最近的功能啟動

下表列出可使用新 Step Functions 的區域。

啟動日期	特徵名稱	可用地區
2023 年 11 月 26 日	呼叫公用 HTTPS 端點並測試個別狀態	• 美國東部 (維吉尼亞北部) – us-east-1 • 美國西部 (奧勒岡) – us-west-2 • 美國東部 (俄亥俄) – us-east-2 • 歐洲 (愛爾蘭) – eu-west-1 • 歐洲 (法蘭克福) – eu-central-1 • 歐洲 (斯德哥爾摩) – eu-north-1 • 亞太區域 (雪梨) – ap-southeast-2 • 亞太區域 (東京) – ap-northeast-1 • 亞太區域 (新加坡) – ap-southeast-1
2023 年 11 月 15 日	Redrive處決	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。
2023 年 10 月 12 日	最佳化的整合 Amazon EMR Serverless	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。

啟動日期	特徵名稱	可用地區
2023年9月07 日	增強的錯誤處理	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。
2023 年 8 月 31 日	工作流程 Studio 增強功能，提供簡化的撰寫	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。
2023 年 6 月 22 日	版本和別名	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。
2023 年 6 月 16 日	全新 AWS SDK 整合	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。
2022年12月01 日	使用「分散式地圖」狀態協調大規模的 parallel 工作流程，以	如需可使用此功能AWS 區域的完整清單，請參閱「地區」下拉式清單中標題為「地區」(Region) 的頁面上AWS 服務的選項。

文件歷史紀錄

本節列出AWS Step Functions 開發人員指南的重大變更。

變更	描述	變更日期
更新	Step Functions 新增對「開啟工作流程」量度 透過開放式工作流程指標，您現在可以看到正在進行的標準工作流程數目以及開啟的工作流程限制的帳戶層級。您可以跨所有工作流程管理工作負載，無論工作流程如何啟動，以確保工作流程運作順暢。您可以設定 CloudWatch 警示來監控工作流程，並在接近限制時主動接收警示。收到警示後，您可以採取動作 (例如停止特定工作流程或要求提高限制)，有效管理工作流程。 開放式工作流程量度可用 CloudWatch 於標準工作流程，無需其他設定。如需進一步了解，請參閱執行指標。	2024 年 2 月 29 日
更新	服務整合新增和更新。如需新增和更新的 AWS SDK 整合清單，請參閱變更支援 AWS SDK 整合的記錄。如需服務的完整清單，請參閱支援的 AWS SDK 服務整合。	2024年1月18日
新功能	使用中的工作流程 Studio Application Composer 來使用AWS CloudFormation範本建置無伺服器工作流程。如需詳細資訊，請參閱 使用工作流程工作室 Application Composer。	2023 年 11 月 27 日
新功能	Step Functions現在可讓您直接呼叫公用 HTTPS 端點，並使用新的測試狀態 API 測試個別狀態。如需詳細資訊，請參閱： • 呼叫第三方 API • 使用 TestState API 來測試狀態	2023 年 11 月 26 日
新功能	Step Functions 現在與 Amazon Bedrock 整合。如需詳細資訊，請參閱下列主題： • 使用 Step Functions 呼叫 Amazon Bedrock • IAM的權限 Amazon Bedrock • 使用執行 AI 提示鏈結 Amazon Bedrock	2023 年 11 月 26 日

變更	描述	變更日期
	• AWS Step Functions 搭配其他服務使用	
新功能	Step Functions 現在可讓您從失敗點開始執行類型為「標準」的redrive工作流程。如需詳細資訊，請參閱 Redriving處決 及 Redriving地圖運行。	2023 年 11 月 15 日
僅文件更新	已發佈新主題，說明如何使用Amazon EventBridge Scheduler. 如需詳細資訊，請參閱 使用亞馬遜 EventBridge 排程器 AWS Step Functions。	2023 年 10 月 16 日
新功能	Step Functions 現在與 Amazon EMR Serverless 整合。如需詳細資訊，請參閱下列主題： • 使用 Step Functions 呼叫 Amazon EMR Serverless • 執行EMR Serverless工作 • Step Functions 的最佳化整合 • AWS Step Functions 搭配其他服務使用	2023 年 10 月 12 日
僅文件更新	已新增有關使用的排程執行狀態機器的相關資訊Amazon EventBridge Scheduler。如需詳細資訊，請參閱 使用 EventBridge 排程器。	2023 年 10 月 5 日
更新	重新組織並更新了「分散式地圖」狀態主題，以提高清晰度，簡潔，並為新用戶建立清晰的旅程圖。如需詳細資訊，請參閱 在分散式模式中使用 Map 狀態來協調大規模的 parallel 工作負載。	2023 年 10 月 6 日
修正項目	修正了教程中使用 AWS CDK v2 的代碼示例。如需詳細資訊，請參閱 建立要Step Functions使用的Lambda狀態機 AWS CDK。	2023 年 9 月 19 日
更新	已新增有關 Step Functions 為清楚識別錯誤而引入的增強錯誤處理功能的相關資訊，並以更好的控制權來實作重試。如需詳細資訊，請參閱 Fail 及 發生錯誤後重試。	2023年9月07 日
更新	Step Functions 已新增工作流程 Studio 的增強功能，以簡化工作流程撰寫體驗。如需詳細資訊，請參閱 AWS Step Functions 流程工作室。	2023 年 8 月 31 日

變更	描述	變更日期
僅文件更新	已新增測量結果報告實際量度計數兩倍的`ExecutionsStarted` 相關資訊。如需詳細資訊，請參閱 報告計數的量度。	2023 年 7 月 25 日
僅文件更新	Step Functions 新增了兩個新的範例專案，這些專案示範了分散式地圖狀態的下列常見使用案例： • 正在處理 CSV 檔案 • 在 Amazon S3 儲存貯體中處理資料	2023 年 7 月 17 日
僅文件更新	發佈有關使用 Terraform 部署狀態機器的新主題。如需詳細資訊，請參閱 使用地形部署狀態機。	2023 年 7 月 5 日
僅文件更新	更新下列程序以配對 Amazon EventBridge 界面的變更。 • 將 Step Functions 事件遞送至 EventBridge • 啟動狀態機器執行以回應 Amazon S3 事件	2023 年 6 月 26 日
新功能	Step Functions 現在提供建立多個狀態機器版本和別名的功能，以提升彈性，同時部署無伺服器工作流程。如需詳細資訊，請參閱 使用版本和別名管理持續部署。	2023 年 6 月 22 日
僅文件更新	改進了`TimeoutSeconds` 和`HeartbeatSeconds` 字段的描述，以描述它們彼此之間的不同。如需詳細資訊，請參閱 工作狀態欄位。	2023 年 6 月 22 日
僅文件更新	發佈新章節，說明如何將通常傳回為「平行」和「對映」狀態傳回的陣列陣列陣列平面化。如需詳細資訊，請參閱 扁平化陣列的陣列。	2023 年 6 月 20 日
更新	Step Functions 透過新增七個 AWS 服務 和 468 個新的 API 動作，擴大了對 AWS SDK 整合的支援。如需詳細資訊，請參閱 支援的 AWS SDK 服務整合 及 變更支援 AWS SDK 整合的記錄。	2023 年 6 月 16 日
僅文件更新	發佈了一個新主題，其 AWS 區域 中列出了最近啟動的 Step Functions 可用。如需詳細資訊，請參閱 最近的功能啟動。	2023 年 6 月 16 日

變更	描述	變更日期
僅文件更新	Step Functions 現在包含一個關於 AWS 服務 的章節 AWS 使用者通知，可作為 AWS 通知的中心位置 AWS Management Console。如需詳細資訊，請參閱 AWS 使用者通知與 Step Functions 搭配使用。	2023 年 5 月 4 日
僅文件更新	新增一節，說明將子工作流程執行結果寫入使用AWS Key Management Service(AWS KMS)金鑰加密的 Amazon S3 儲存貯體所需的許可。如需詳細資訊，請參閱 AWS KMS key 加密 Amazon S3 儲存貯體的 IAM 許可。	2023 年 4 月 25 日
僅文件更新	已新增說明資料流程模擬器功能的新主題。如需詳細資訊，請參閱 數據流模擬器。	2023 年 4 月 14 日
配額更新	已新增有關每個帳戶中開啟地圖執行預設配額 1000 的資訊。如需詳細資訊，請參閱 與帳戶相關的配額。	2023年4月05 日
僅文件更新	已新增說明何時將 AWS Data Pipeline 工作負載移轉至 Step Functions 的主題。本主題也提供說明如何執行移轉的範例清單。如需詳細資訊，請參閱 將工作負載從移轉 AWS Data Pipeline 至 Step Functions。	2023 年 3 月 30 日
僅文件更新	已新增有關「分散式貼圖」狀態中無法使用 X-Ray 追蹤的備註。如需詳細資訊，請參閱 AWS X-Ray 和 Step Functions。	2023 年 3 月 21 日
僅文件更新	已新增有關 Step Functions 式如何處理標籤授權的資訊。如需詳細資訊，請參閱 步驟函數中的標記 及 標籤類型政策。	2023 年 3 月 15 日
僅文件更新	已新增有關 Step Functions 如何剖析用作分散式對應狀態中輸入的 CSV 檔案的資訊。如需詳細資訊，請參閱 Amazon S3 儲存貯體中的 CSV 檔案。	2023 年 3 月 14 日
僅文件更新	已新增有關 Step Functions 式如何處理執行 Job (.sync) 模式之跨帳戶呼叫的相關資訊。如需詳細資訊，請參閱執行 Job (.sync)。	2023年3月01 日
僅文件更新	將已完成工作流程執行的歷程記錄保留期從 90 天縮短為 30 天。如需調整保留期間的詳細資訊，請參閱執行保證和與狀態機器執行相關的配額。	2023 年 2 月 21 日

變更	描述	變更日期
更新	Step Functions 透過新增 35 項 AWS 服務和 1100 個新的 API 動作，擴大了對 AWS SDK 整合的支援。如需詳細資訊，請參閱 支援的 AWS SDK 服務整合 及 變更支援 AWS SDK 整合的記錄。	2023 年 2 月 17 日
僅文件更新	發佈入門教學課程系列，引導您完成使用 Step Functions 數建立信用卡申請工作流程的程序。如需詳細資訊，請參閱 AWS Step Functions 入門。	2022 年 12 月 30 日
新功能	Step Functions 增加了使用新的分散式Map狀態模式來協調大規模 parallel 工作流程的支援，以進行資料處理。如需詳細資訊，請參閱 在分散式模式中使用 Map 狀態來協調大規模的 parallel 工作負載。	2022年12月01日
新功能	Step Functions 現在支援存取其他帳戶中設定的跨帳戶 AWS 資源。如需詳細資訊，請參閱 • 存取工作流程 AWS 帳戶 中其他資源 • 教學課程：存取跨帳戶 AWS 資源 • Task state	2022 年 11 月 18 日
更新	Step Functions 現在提供檢視和偵錯 Express 工作流程執行的全新主控台體驗。如需詳細資訊，請參閱: • 主控台中的標準和快速工作流程執行 • 在 Step Functions 主控台上檢視和偵錯執行	2022 年 10 月 18 日
更新	新增支援，可在使用addStep和 addStep.sync API 進行 Amazon EMR 優化服務整合時選擇性地指定ExecutionRoleArn 參數。如需詳細資訊，請參閱 使用 Step Functions 呼叫 Amazon EMR。	2022 年 9 月 20 日
僅文件更新	新增一個新主題，提供有關使用 Step Functions 建置無伺服器工作流程時最佳化成本的建議。如需詳細資訊，請參閱 使用快速工作流程最佳化。	2022 年 9 月 15 日

變更	描述	變更日期
更新	Step Functions 增加了對 14 個新的內在函數的支援，用於執行資料處理工作，例如陣列操作、資料編碼和解碼、雜湊計算、JSON 資料操作、數學函數運算和唯一識別碼產生。 僅文檔更新： 根據它們幫助您執行的數據處理任務的類型，將所有現有和新引入的內部函數分組為以下類別： • 數組的內在函數 • 數據編碼和解碼的內在函數 • 散列計算的內在 • JSON 數據操作的內在函數 • 數學運算的內在函數 • 字符串操作的內在 • 唯一標識符生成的內在 • 泛型操作的內在 如需詳細資訊，請參閱 內部函數。	2022 年 8 月 31 日
更新	Step Functions 增加了另外三個 AWS 服務，擴大了對 AWS SDK 整合的支援 — AWS Billing Conductor Amazon GameSparks、和Amazon Pinpoint SMS and Voice V2。如需詳細資訊，請參閱 變更支援 AWS SDK 整合的記錄。	2022 年 7 月 26 日
僅文件更新	新增了一個新主題，其中包含對 Step Functions 支援的 AWS SDK 整合所做的所有更新摘要。如需更多資訊，請參閱變更支援 AWS SDK 整合的記錄	2022 年 7 月 26 日
僅文件更新	AWS Step Functions 開發人員指南現在包含特別針對 Express 工作流程發出的執行指標的詳細資料。如需詳細資訊，請參閱 快速工作流程的執行度量。	2022年6月09 日

變更	描述	變更日期
更新	Step Functions 主控台增強 主控台現在具有重新設計的 [執行詳細資料] 頁面，其中包含下列增強 • 能夠一目了然地識別執行失敗的原因。 • 狀態機的兩種新視覺化模式 — 「表格檢視」 和「事件」檢視。這些檢視也可讓您套用篩選器，以僅檢視感興趣的資訊。此外，您可以根據事件時間戳記對「事件」檢視內容進行排序。 • 使用下拉式清單或在「表格」檢視模式的**Map**狀態樹狀檢視中，在「圖形」檢視模式的不同Map狀態迭代之間切換。 • 檢視有關工作流程中每個狀態的深入資訊，包括完整的輸入和輸出資料傳輸路徑，以及Task或Parallel狀態的重試嘗試。 • 其他增強功能包括複製狀態機器的執行 Amazon Resource Name、檢視狀態機器轉換總計數，以及以 JSON 格式匯出執行詳細資訊的選項。 僅文件更新 已新增主題，說明「執行詳細資訊」頁面中顯示的各種資訊類型。此外，還加入了一個教學課程來展示如何檢查此資訊。如需詳細資訊，請參閱： • 在 Step Functions 主控台上檢視和偵錯執行 • 教學課程：使用 Step Functions 主控台檢查狀態機器執行	2022年5月9日
更新	Step Functions 現在提供了一種因應措施，以防止混淆的副安全性問題，當實體 (服務或帳戶) 被不同的實體強制執行動作時，就會產生混淆的安全性問題。如需詳細資訊，請參閱： • 防止跨服務混淆副問題	2022年5月02 日

變更	描述	變更日期

更新	• Step Functions 通過添加 21 個 AWS 服務擴展了對 AWS SDK 集成的支持。如需詳細資訊，請參閱 支援的 AWS SDK 服務整合。
	• 僅文件更新：
	  • 已新增當您錯誤地執行 AWS SDK 服務與 Step Functions 整合時所產生之例外狀況前置字元的所有例外狀況前置字元清單。如需詳細資訊，請參閱 支援的 AWS SDK 服務整合。
	  • 針對支援的 AWS SDK 整合新增了所有不受支援 API 動作的清單。如需詳細資訊，請參閱 支援服務不支援的 API 動作。
	  • 已新增目前已停用的所有受支援 AWS SDK 整合的清單。如需詳細資訊，請參閱 已淘汰的 AWS SDK 服務整合。
		2022 年 4 月 19 日

新功能	Step Functions 本地現在支持 AWS SDK 集成和模擬服務集成。如需詳細資訊，請參閱： • 使用模擬服務集成	2022 年 1 月 28 日
新功能	AWS Step Functions 現在支援建立具有同步快速狀態機器的 Amazon API Gateway REST API 作為使用 AWS Cloud Development Kit (AWS CDK). 如需詳細資訊，請參閱： • 使用同步快速狀態機器建立 API Gateway REST API AWS CDK	2021 年 12 月 10 日
更新	Step Functions 新增了三個新的範例專案，這些專案示範了 Step Functions 和 Amazon Athena 升級後的主控台的整合。如需詳細資訊，請參閱： • 執行多個查詢（Amazon Athena，Amazon SNS） • 查詢大型資料集 (Amazon Athena、Amazon S3 AWS Glue、Amazon SNS) • 讓資料保持最新狀態 (Amazon Athena、Amazon S3 AWS Glue)	2021 年 11 月 22 日
新功能	Step Functions 新增了對同步快速工作流程的 Amazon VPC 端點支援。如需詳細資訊，請參閱： • 適用於 Step Functions 的 Amazon VPC 端點	2021 年 11 月 15 日

變更	描述	變更日期
更新	AWS Step Functions 已新增三個新的範例專案，示範如何使用 Step Functions AWS Batch 整合。如需詳細資訊，請參閱： • 扇出一 AWS Batch 份工作 • AWS Batch 與 Lambda • 使用Step Functions和AWS Batch錯誤處理	2021 年 10 月 14 日
新功能	AWS Step Functions 已新增 AWS SDK 整合功能，可讓您針對超過二百項 AWS 服務使用 API 動作。如需詳細資訊，請參閱： • AWS SDK 服務整合 • 使用 AWS SDK 服務整合收集 Amazon S3 儲存貯體資訊	2021 年 9 月 30 日
新功能	AWS Step Functions 已加入視覺化工作流程設計器，即 AWS Step Functions 工作流程 Studio。如需詳細資訊，請參閱： • AWS Step Functions 流程工作室 • 學習如何使用 AWS Step Functions 工作流程工作室	2021 年 6 月 17 日
更新	AWS Step Functions 在 CodeBuild 整合中新增了四個新 API `DeleteBuildBatch` 、、`RetryBuildBatch` 和。`StartBuildBatch` `StopBuildBatch` 如需詳細資訊，請參閱： • AWS CodeBuild 使用 Step Functions 調用	2021 年 6 月 4 日
新功能	AWS Step Functions 現在與 Amazon 集成 EventBridge。如需詳細資訊，請參閱： • EventBridge 使用 Step Functions 調用 • Step Functions 和 IAM 政策 Amazon 的 IAM 政策 EventBridge • 範例專案，展示如何 將自訂事件傳送至 EventBridge	2021 年 5 月 14 日

變更	描述	變更日期
更新	AWS Step Functions 已新增一個新的範例專案，顯示如何使用 Step Functions 數和 Amazon Redshift 資料 API 來執行 ETL/ELT 工作流程。如需詳細資訊，請參閱： • 使用亞馬遜紅移 (Lambda、亞馬 Amazon Redshift 資料 API) 執行 ETL/ELT 工作流程	2021 年 4 月 16 日
新功能	AWS Step Functions 在控制台中有一個新的數據流模擬器。如需詳細資訊，請參閱： • Step Functions 控制台	2021 年 4 月 8 日
新功能	AWS Step Functions 現在與 EKS 上的 Amazon EMR 集成。如需詳細資訊，請參閱： • 在 EKS 上致電 Amazon EMR AWS Step Functions	2021 年 3 月 29 日
更新	狀態機器定義的 YAML 支援已新增至 AWS Toolkit for Visual Studio Code 和 AWS CloudFormation。如需詳細資訊，請參閱： • 定義格式支援 • AWS Toolkit for Visual Studio Code	2021 年 3 月 4 日
新功能	AWS Step Functions 現在整合了 AWS Glue DataBrew. 如需詳細資訊，請參閱： • 使用 Step Functions 管理 AWS Glue DataBrew 工作 • 什麼是 AWS Glue DataBrew？ 在 DataBrew 開發人員指南中。	2021 年 1 月 6 日
新功能	AWS Step Functions 現在可以使用同步 Express 工作流程，讓您輕鬆協調微服務。如需詳細資訊，請參閱： • 同步和非同步快速工作流 • 範例專案，展示如何 叫用同步快速工作流 • StartSyncExecution應用程式介面文件。	2020 年 11 月 24 日

變更	描述	變更日期
新功能	AWS Step Functions 現在與 Amazon API Gateway 集成。如需詳細資訊，請參閱： • 使用 Step Functions 呼叫 API Gateway • Step Functions 和 IAM 政策 Amazon API Gateway 的 IAM 政策 • 範例專案，展示如何 呼叫 API Gateway	2020 年 11 月 17 日
新功能	AWS Step Functions 現在與 Amazon Elastic Kubernetes Service 集成。如需詳細資訊，請參閱： • 使用 Step Functions 調用 Amazon EKS • Step Functions 和 IAM 政策 Amazon EKS 的 IAM 政策 • 範例專案，展示如何 管理 Amazon EKS 叢集	2020 年 11 月 16 日
新功能	AWS Step Functions 現在與 Amazon Athena 集成。如需詳細資訊，請參閱： • 使用 Step Functions 呼叫 Athena • Step Functions 和 IAM 政策 Amazon Athena 的 IAM 政策 • 範例專案，展示如何 開始 Athena 查詢	2020 年 10 月 22 日
新功能	AWS Step Functions 現在支援使用追蹤 end-to-end 工作流程 AWS X-Ray，讓您能夠完全掌握狀態機器執行情況，並讓分散式應用程式的分析和偵錯變得更加容易。如需詳細資訊，請參閱： • AWS X-Ray 和 Step Functions • Step Functions 和 IAM 政策 的 IAM 政策 AWS X-Ray • AWS Step Functions API 參考 • TracingConfiguration	2020 年 9 月 14 日

變更	描述	變更日期

更新	AWS Step Functions 現在支援以 UTF-8 編碼字串形式，承載大小最多 256 KB 的資料。這可讓您在標準和快速工作流程中處理較大的承載。
	
	您不需要變更現有的狀態機器，即可使用較大的承載。不過，您必須更新至最新版本的 Step Functions 式 SDK 和本機執行程式，才能使用更新的 API。如需詳細資訊，請參閱：
	
	• 配額
	• the section called “使用 Amazon S3 ARN 而不是傳遞大型有效載荷”
	• `States.DataLimitExceeded`
	• the section called “CloudWatch記錄有效載荷”
	• the section called “EventBridge 有效載荷”
	• AWS Step Functions API 參考
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
	  • StateEnteredEventDetails
	  • StateExitedEventDetails
	  • TaskSubmittedEventDetails
	  • TaskSucceededEventDetails
		2020 年 9 月 3 日

變更	描述	變更日期

更新	Amazon 州語言已更新如下：
	• 選擇規則已添加
	  • 空比較運算子、IsNull。 IsNull針對 JSON 空值進行測試，並且可用於檢測先前狀態的輸出是否為空。
	  • 已新增、、、 IsBoolean和其他四個新運算 IsString 子 IsTimestamp。 IsNumeric
	  • 使用IsPresent 運算符對字段的存在或不存在的測試。IsPresent 可用來防止嘗試存取不存在的金鑰時發States.Runtime 生錯誤。
	  • 萬用字元模式比對，可支援字串比較具有一或多個萬用字元的模式。
	  • 支持的比較運算符兩個變量之間的比較。
	• 現在可以從Task狀態輸入動態提供逾時和活動訊號值，而不是使用TimeoutSecondsPath 和HeartbeatSecondsPath 欄位的固定值。如需詳細資訊，請參閱任務狀態。
	• 新ResultSelector欄位提供了一種在套用狀態之前ResultPath 操作狀態結果的方法。ResultSelector 欄位是Map、平行和任務狀態中的選擇性欄位。
	• 內部函數已被添加到允許沒有Task狀態的基本操作。內建函數可以在Parameters 和ResultSelector 欄位中使用。
		2020 年 8 月 13 日

更新	AWS Step Functions 現在支援 Amazon SageMaker CreateProcessingJob API 呼叫。如需詳細資訊，請參閱： • SageMaker 使用 Step Functions 管理 • 預先處理資料並訓練機器學習模型，示範的範例專案CreateProcessingJob 。	2020 年 8 月 4 日

變更	描述	變更日期
新功能	AWS Step Functions 現在受到支援 AWS Serverless Application Model，可讓您更輕鬆地將工作流程協調整合到無伺服器應用程式中。如需詳細資訊，請參閱： • AWS Step Functions 而且 AWS SAM • AWS::Serverless::StateMachine • AWS SAM 政策範本	2020 年 5 月 27 日
新功能	AWS Step Functions 為嵌套步驟函數執行引入了新的同步調用。新的叫用 `arn:aws:states:::states:startExecution.sync:2` 會傳回一個 JSON 物件。原始叫用 `arn:aws:states:::states:startExecution.sync` 會繼續受到支援，並傳回一個 JSON 逸出字符。如需詳細資訊，請參閱： • 以整合式服務 AWS Step Functions 的形式管理執行	2020 年 5 月 19 日
新功能	AWS Step Functions 現在整合了 AWS CodeBuild. 如需詳細資訊，請參閱： • AWS Step Functions 搭配其他服務使用 • AWS CodeBuild 使用 Step Functions 調用 • Step Functions 的最佳化整合	2020 年 5 月 5 日
新功能	中現在支援 Step Functions AWS Toolkit for Visual Studio Code，讓您無需離開程式碼編輯器即可輕鬆建立和視覺化狀態機型工作流程。	2020 年 3 月 31 日
更新	您現在可以為標準工作流程設定 Amazon CloudWatch 日誌記錄。如需詳細資訊，請參閱： • 記錄使用CloudWatch日誌	2020 年 2 月 25 日
新功能	AWS Step Functions 現在可以直接從 Amazon 虛擬私有雲 (VPC) 存取，而不需要公有 IP 位址。如需詳細資訊，請參閱： • 適用於 Step Functions 的 Amazon VPC 端點	2019 年 12 月 23 日

變更	描述	變更日期

新功能	快速工作流程是新的工作流程類型，適用於大量事件處理工作負載，例如 IoT 資料擷取、串流資料處理和轉換，以及行動應用程式後端。
	
	如需詳細資訊，請檢閱下列新增和更新的主題。
	
	• 標準與快速工作流程
	  • 執行保證
	• AWS Step Functions 搭配其他服務使用
	  • Step Functions 的最佳化整合
	• 處理來自 Amazon SQS 的大量訊息 (快速工作流程)
	• 選擇性檢查點範例 (快速工作流程)
	• 配額
	  • 配額
	• 記錄使用CloudWatch日誌
	• AWS Step Functions API 參考
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
	  • ListExecutions
	  • ListStateMachines
	  • StartExecution
	  • CloudWatchLogsLogGroup
	  • LogDestination
	  • LoggingConfiguration
		2019 年 12 月 3 日

變更	描述	變更日期
新功能	AWS Step Functions 現在與 Amazon EMR 集成。如需詳細資訊，請參閱： • AWS Step Functions 搭配其他服務使用 • 使用 Step Functions 呼叫 Amazon EMR • Step Functions 的最佳化整合	2019 年 11 月 19 日

更新	AWS Step Functions 已經發布了 AWS Step Functions 數數據科學 SDK。如需更多資訊，請參閱下列內容。
	• Github 上的專案
	• 開發套件文件
	• 下列範例筆記本，可在SageMaker主控台和相關GitHub 專案中使用。
	  • `hello_world_workflow.ipynb`
	  • `machine_learning_workflow_abalone.ipynb`
	  • `training_pipeline_pytorch_mnist.ipynb`
		2019 年 11 月 7 日

更新	Step Functions 現在支援適用於 Amazon 的更多 API 動作 SageMaker，並包含兩個新的範例專案來展示功能。如需更多資訊，請參閱下列內容。 • SageMaker 使用 Step Functions 管理 • AWS Step Functions 搭配其他服務使用 • 訓練機器學習模型 • 調校機器學習模型	2019 年 10 月 3 日
新功能	Step Functions 支援以整合式服務 API 呼叫`StartExecution` 來啟動新的工作流程執行。請參閱： • 從任務狀態開始工作流程執行 • 以整合式服務 AWS Step Functions 的形式管理執行 • AWS Step Functions 搭配其他服務使用 • 啟動 Step Functions 工作流程執行的 IAM 政策	2019 年 8 月 12 日

變更	描述	變更日期
新功能	Step Functions 包括將任務 Token 傳遞給整合式服務，並暫停執行，直到使用`SendTaskSuccess` 或傳回該工作 Token 為止`SendTaskFailure` 。請參閱： • 服務整合模式 • 等候傳回任務字符的回呼 • 回呼模式範例 (Amazon SQS、Amazon SNS、Lambda) • Step Functions 的最佳化整合 • 部署人工核准專案範例 • 服務整合指標 Step Functions 現在提供了一種直接在狀態定義`"Parameters"` 字段中訪問有關當前執行的動態信息的方法。請參閱： • 內容物件 • 依參數方式傳遞內容物件節點	2019 年 5 月 23 日
新功能	Step Functions 支持執行狀態更改的 CloudWatch 事件，請參閱： • EventBridge (CloudWatch 事件) 用於 Step Functions 執行狀態變更 • Amazon CloudWatch 活動用戶指南	2019 年 5 月 8 日
新功能	Step Functions 支援使用標籤的 IAM 許可。如需詳細資訊，請參閱： • 步驟函數中的標記 • 標籤類型政策	2019 年 3 月 5 日
新功能	Step Functions 本地現在可用。您可以在本地計算機上運行 Step Functions 進行測試和開發。Step Functions 本地可供下載作為 Java 應用程序，或作為碼頭圖像。請參閱在本地測試狀態機。	2019 年 2 月 4 日
新功能	AWS Step Functions 現已在北京和寧夏地區推出。請參閱支援的 區域。	2018 年 1 月 15 日

變更	描述	變更日期
新功能	Step Functions 支援資源標記，以協助追蹤您的成本分配。您可以在 Details (詳細資訊) 頁面或透過 API 動作標記狀態機器。請參閱步驟函數中的標記。	2019 年 1 月 7 日
新功能	AWS Step Functions 現已在歐洲 (巴黎) 和南美洲 (聖保羅) 地區推出。請參閱支援的 區域。	2018 年 12 月 13 日
新功能	AWS Step Functions 歐洲 (斯德哥爾摩) 地區現已推出。如需支援的區域清單，請參閱支援的 區域。	2018 年 12 月 12 日
新功能	Step Functions 現在與某些 AWS 服務集成。您現在可以從 Amazon 州語言的任務狀態直接呼叫這些整合服務的 API，並將參數傳遞至這些整合服務的 API。如需詳細資訊，請參閱： • AWS Step Functions 搭配其他服務使用 • 將參數傳遞至服務 API • Step Functions 的最佳化整合	2018 年 11 月 29 日
更新	改進任務狀態文件中 `TimeoutSeconds` 和 `HeartbeatSeconds` 的說明。請參閱任務。	2018 年 10 月 24 日
更新	改進執行歷史記錄大小上限限制的說明內容，並提供相關最佳實務主題的連結。 • 與狀態機器執行相關的配額 • 避免達到歷史記錄配額	2018 年 10 月 17 日
更新	添加了一個新的教程的 AWS Step Functions 文檔：請參閱啟動狀態機器執行以回應 Amazon S3 事件。	2018 年 9 月 25 日

變更	描述	變更日期
更新	從限制文件移除在 Step Functions 主控台中顯示的最大執行項目。請參閱配額。	2018 年 9 月 13 日
更新	在文件中新增最佳作法主題， AWS Step Functions 說明在輪詢活動工作時改善延遲。請參閱輪詢活動任務時避免延遲。	2018 年 8 月 30 日
更新	改進了有 AWS Step Functions 關活動和活動工作者的主題。請參閱活動。	2018 年 8 月 29 日
更新	改進了有關 CloudTrail 集成的 AWS Step Functions 主題。請參閱記錄步驟函數使用 AWS CloudTrail。	2018 年 8 月 7 日
更新	在 AWS CloudFormation 教程中添加了 JSON 示例。請參閱使用建立 Step Functions 的 Lambda 狀態機 AWS CloudFormation。	2018 年 6 月 23 日
更新	已新增有關處理 Lambda 服務錯誤的新主題。請參閱處理 Lambda 務例外。	2018 年 6 月 20 日
新功能	AWS Step Functions 亞太區域 (孟買) 區域現已推出。如需支援的區域清單，請參閱支援的 區域。	2018 年 6 月 28 日
新功能	AWS Step Functions AWS GovCloud (美國西部) 區域現已推出。如需支援的區域清單，請參閱支援的 區域。如需有關在 AWS GovCloud (美國西部) 區域中使用 Step Functions 的資訊，請參閱AWS GovCloud (US)。	2018 年 6 月 28 日
更新	改善 `Parallel` 狀態錯誤處理的文件。請參閱錯誤處理。	2018 年 6 月 20 日

變更	描述	變更日期
更新	改進了有關 Step Functions 中輸入和輸出處理的文檔。了解如何使用 `InputPath` 、`ResultPath` 及 `OutputPath` 以透過您的工作流程、狀態和任務來控制 JSON 的流程。請參閱： • Step Functions 中的輸入和輸出處理 • ResultPath	2018 年 6 月 7 日
更新	改善 Parallel 狀態的程式碼範例。請參閱平行。	2018 年 4 月 6 日
新功能	您現在可以在中監控 API 和服務指標 CloudWatch。請參閱監視 Step Functions 使用 CloudWatch。	2018 年 5 月 25 日
更新	現已提升了 `StartExecution` 、`StopExecution` 及 `StateTransition` 在以下區域的調節限制： • 美國東部 (維吉尼亞北部) • 美國西部 (奧勒岡) • 歐洲 (愛爾蘭) 如需更多資訊，請參閱配額。	2018 年 5 月 16 日
新功能	AWS Step Functions 美國西部 (加利佛尼亞北部) 和亞太區域 (首爾) 區域現已推出。如需支援的區域清單，請參閱支援的 區域。	2018 年 5 月 5 日
更新	更新程序和映像，以符合界面的變更。	2018 年 4 月 25 日
更新	新增教學，說明如何啟動新執行以繼續您的工作。請參閱將長時間執行的工作流程執行作為新的執行作業。本教學會說明可以避開一些服務限制的設計模式。請參閱避免達到歷史記錄配額。	2018 年 4 月 19 日
更新	透過新增有關狀態機器的概念資訊，改善狀態文件的簡介。請參閱狀態。	2018 年 3 月 9 日

變更	描述	變更日期
更新	除了 HTML、PDF 和 Kindle 之外，《 AWS Step Functions 開發人員指南》也可在上 GitHub取得。若要留下信用評價，請選擇右上角的 GitHub 圖示。 	2018 年 3 月 2 日
更新	已新增描述與 Step Functions 相關之其他資源的主題。 請參閱相關資訊。	2018 年 2 月 20 日

新功能	• 當您建立新的狀態機器時，您必須確認 AWS Step Functions 將建立可存取 Lambda 函數的 IAM 角色。
	• 更新以下教學，以反映狀態機器建立工作流程的微幅變更：
	  • 建立使用 Lambda 的 Step Functions 狀態機器
	  • 使用 Step Functions 創建活動狀態機
	  • 使用 Step Functions 狀態機處理錯誤條件
	  • 用 Lambda 迭代一個循環
		2018 年 2 月 19 日

更新	新增主題，其說明以 Ruby 編寫的範例活動工作者。此實作可以用來直接建立 Ruby 活動工作者，或做為建立其他語言活動工作者的設計模式。 請參閱Ruby 中的範例活動工作者。	2018 年 2 月 6 日
更新	新增了一個新的教學課程，說明使用 Lambda 函數迭代計數的設計模式。 請參閱建立使用 Lambda 的 Step Functions 狀態機器。	2018 年 1 月 31 日
更新	已更新要包含`DescribeStateMachineForExecution` 和 `UpdateStateMachine` API 的 IAM 許可的內容。 請參閱為非管理員使用者建立精細的 IAM 許可。	2018 年 1 月 26 日

變更	描述	變更日期
更新	新增可用地區：加拿大 (中部)、亞太區域 (新加坡)。 請參閱支援的 區域。	2018 年 1 月 25 日
更新	更新了教學課程和程序，以反映 IAM 允許您選擇 Step Functions 作為角色。	2018 年 1 月 24 日
更新	新增最佳實務主題，其會建議不要在狀態之間傳遞大型承載。 請參閱使用 Amazon S3 ARN 而不是傳遞大型有效載荷。	2018 年 1 月 23 日
更新	更正程序以符合建立狀態機器的界面更新： • 建立使用 Lambda 的 Step Functions 狀態機器 • 使用 Step Functions 創建活動狀態機 • 使用 Step Functions 狀態機處理錯誤條件	2018 年 1 月 17 日
新功能	您可以使用範例專案來快速佈建狀態機器和所有相關 AWS 資源。請參閱Step Functions 的範例專案， 可用的範例專案包括： • 投票 Job 狀態 (Lambda、 AWS Batch) • 任務計時器（Lambda，Amazon SNS） Note 這些範例專案和相關文件會取代說明實作相同功能的教學。	2018 年 1 月 11 日
更新	新增最佳實務章節，其中包含有關避免停滯執行的資訊。請參閱Step Functions 的最佳做法。	2018 年 1 月 5 日

變更	描述	變更日期
更新	已新增重試如何影響定價的備註： **Note** 重試被視為狀態轉換。有關狀態轉換如何影響計費的詳細資訊，請參閱 Step Functions 定價。	2017 年 12 月 8 日
更新	新增與資源名稱相關的資訊： **Note** Step Functions 可讓您建立狀態機器、執行項目和活動的名稱，以及包含非 ASCII 字元的標籤。這些非 ASCII 名稱不適用於 Amazon CloudWatch。若要確保您可以追蹤 CloudWatch 量度，請選擇僅使用 ASCII 字元的名稱。	2017 年 12 月 6 日
更新	改善安全性概觀資訊，並新增有關精細 IAM 許可的主題。請參閱 中的安全性 AWS Step Functions 和 為非管理員使用者建立精細的 IAM 許可。	2017 年 11 月 27 日
新功能	您可以更新現有的狀態機器。請參閱更新您的狀態機。	2017 年 11 月 15 日

變更	描述	變更日期
更新	新增可釐清 `Lambda.Unknown` 錯誤的備註和在以下章節前往 Lambda 文件的連結： • 錯誤名稱 • 步驟 3：使用 Catch 欄位建立狀態機 **Note** Lambda 中未處理的錯誤會報告為錯誤輸出`Lambda.Unknown` 中。這些包括 out-of-memory 錯誤和功能超時。您可以在`Lambda.Unknown` `States.ALL` 、或上進行比對`States.TaskFailed` 來處理這些錯誤。當 Lambda 達到調用的最大數量時，錯誤為。`Lambda.TooManyRequestsException` 如需 Lambda 函數錯誤的詳細資訊，請參閱AWS Lambda 開發人員指南中的錯誤處理和自動重試。	2017 年 10 月 17 日
更新	更正並澄清 IAM 指示，並更新了所有教學課程中的螢幕擷取畫面。	2017 年 10 月 11 日

更新	• 已新增狀態機器執行結果的螢幕擷取畫面，以反映 Step Functions 主控台中的變更。重新撰寫下列教學課程中的 Lambda 指示，以反映 Lambda 主控台中的變更：
	  • 建立使用 Lambda 的 Step Functions 狀態機器
	  • 建立工作狀態輪詢器
	  • 建立任務計時器
	  • 使用 Step Functions 狀態機處理錯誤條件
	• 更正並釐清以下章節有關建立狀態機器的資訊：
	  • 使用 Step Functions 創建活動狀態機
		2017 年 10 月 6 日

變更	描述	變更日期
更新	重新撰寫下列各節中的 IAM 指示，以反映 IAM 主控台中的變更： • 為狀態機建立 IAM 角色 • 建立使用 Lambda 的 Step Functions 狀態機器 • 建立工作狀態輪詢器 • 建立任務計時器 • 使用 Step Functions 狀態機處理錯誤條件 • 使用 API Gateway 建立 Step Functions 式 API	2017 年 10 月 5 日
更新	重寫狀態機器資料章節。	2017 年 9 月 28 日
新功能	所有可使用 Step Functions 的區域，都會增加與 API 動作節流相關的限制。	2017 年 9 月 18 日
更新	• 更正並釐清所有教學中有關啟動新執行的資訊。 • 更正並釐清與帳戶相關的配額章節中的資訊。	2017 年 9 月 14 日
更新	重寫下列教學課程，以反映 Lambda 主控台中的變更： • 建立使用 Lambda 的 Step Functions 狀態機器 • 使用 Step Functions 狀態機處理錯誤條件 • 建立工作狀態輪詢器	2017 年 8 月 28 日
新功能	Step Functions 在歐洲（倫敦）可用。	2017 年 8 月 23 日
新功能	狀態機器的視覺化工作流程可讓您放大、縮小和置中圖表。	2017 年 8 月 21 日

<table>
<tr><th>變更</th><th>描述</th><th>變更日期</th></tr>
<tr><td>新功能</td><td>
Important
執行在 90 天內無法使用另一個執行的名稱。

當您使用相同的名稱進行多個 <code>StartExecution</code> 呼叫，新執行不會執行且適用於以下規則。
<table>
<tr><th rowspan="2">輸入類型</th><th colspan="2">執行狀態</th></tr>
<tr><th>開啟</th><th>Closed</th></tr>
<tr><td>Identical (相同)</td><td><code>Success</code></td><td><code>Executio</code>
<code>AlreadyE</code>
<code>ists</code></td></tr>
<tr><td>不同</td><td><code>Executio</code>
<code>AlreadyE</code>
<code>ists</code></td><td><code>Executio</code>
<code>AlreadyE</code>
<code>ists</code></td></tr>
<tr><td colspan="3">如需詳細資訊，<code>name</code>請參閱 API 參考中 <code>StartExecution</code> API 動作的 AWS Step Functions 要求參數。</td></tr>
</table>
</td><td>2017 年 8 月 18 日</td></tr>
<tr><td>更新</td><td>在 使用 API Gateway 建立 Step Functions 式 API 教學中新增有關傳遞狀態機器 ARN 替代方法的資訊。</td><td>2017 年 8 月 17 日</td></tr>
<tr><td>更新</td><td>新增建立工作狀態輪詢器教學。</td><td>2017 年 8 月 10 日</td></tr>
<tr><td>新功能</td><td>

Step Functions 發出<code>ExecutionThrottled</code> CloudWatch 指標。如需詳細資訊，請參閱 監視 Step Functions 使用 CloudWatch。
新增與狀態節流有關的配額章節。

</td><td>2017 年 8 月 3 日</td></tr>
</table>

變更	描述	變更日期
更新	更新了 步驟 1：建立 API Gateway 的 IAM 角色部分中的说明。	2017 年 7 月 18 日
更新	更正並釐清Choice章節中的資訊。	2017 年 6 月 23 日
更新	新增有關在下列教學課程中使用其他 AWS 帳號下資源的資訊： • 建立使用 Lambda 的 Step Functions 狀態機器 • 使用建立 Step Functions 的 Lambda 狀態機 AWS CloudFormation • 使用 Step Functions 創建活動狀態機 • 使用 Step Functions 狀態機處理錯誤條件	2017 年 6 月 22 日
更新	更正並釐清下列章節中的資訊： • 使用 Step Functions 狀態機處理錯誤條件 • 狀態 • Step Functions 中的錯誤處理	2017 年 6 月 21 日
更新	重寫了所有教學課程，以符合 Step Functions 主控台重新整理。	2017 年 6 月 12 日
新功能	Step Functions 適用於亞太區域 (雪梨)。	2017 年 6 月 8 日
更新	重組Amazon States Language章節。	2017 年 6 月 7 日
更新	更正並釐清使用 Step Functions 創建活動狀態機章節中的資訊。	2017 年 6 月 6 日
更新	更正使用重試和使用 Catch 的狀態機器示例章節中的程式碼範例。	2017 年 6 月 5 日

變更	描述	變更日期
更新	使用 AWS 文件標準重新架構本指南。	2017 年 5 月 31 日
更新	更正並釐清平行章節中的資訊。	2017 年 5 月 25 日
更新	將路徑和篩選條件章節合併至Step Functions 中的輸入和輸出處理章節。	2017 年 5 月 24 日
更新	更正並釐清監視 Step Functions 使用 CloudWatch章節中的資訊。	2017 年 5 月 15 日
更新	更新 使用 Step Functions 創建活動狀態機 教學中的 `GreeterActivities.java` 工作者程式碼。	2017 年 5 月 9 日
更新	將簡介影片新增至什麼是 AWS Step Functions？章節。	2017 年 4 月 19 日
更新	更正並釐清下列教學中的資訊： • 建立使用 Lambda 的 Step Functions 狀態機器 • 使用 Step Functions 創建活動狀態機 • 使用 Step Functions 狀態機處理錯誤條件	2017 年 4 月 19 日
更新	已將 Lambda 範本的相關資訊新增至建立使用 Lambda 的 Step Functions 狀態機器和使用 Step Functions 狀態機處理錯誤條件教學課程。	2017 年 4 月 6 日
更新	變更「最大輸入或結果資料大小」限制至「任務、狀態或執行的最大輸入或結果資料大小」(32,768 個字元)。如需詳細資訊，請參閱 與工作執行相關的配額。	2017 年 3 月 31 日

變更	描述	變更日期
新功能	• Step Functions 透過將步 Step Functions 數設定為 Amazon CloudWatch 事件目標來支援執行狀態機器。	2017 年 3 月 21 日
新功能	• Step Functions 數允許 Lambda 函數錯誤處理作為首選的錯誤處理方法。 • 更新使用 Step Functions 狀態機處理錯誤條件教學和Step Functions 中的錯誤處理章節。	2017 年 3 月 16 日
新功能	Step Functions 可在歐洲 (法蘭克福) 使用。	2017 年 3 月 7 日
更新	重新編排目錄中的主題，並更新以下教學： • 建立使用 Lambda 的 Step Functions 狀態機器 • 使用 Step Functions 創建活動狀態機 • 使用 Step Functions 狀態機處理錯誤條件	2017 年 2 月 23 日
新功能	• 「Step Functions」主控台的「狀態機器」頁面包含「複製到新的」和「刪除」按鈕。 • 更新螢幕擷取畫面以符合主控台的變更。	2017 年 2 月 23 日
新功能	• Step Functions 支援使用 API Gateway 建立 API。 • 新增使用 API Gateway 建立 Step Functions 式 API教學課程。	2017 年 2 月 14 日
新功能	• Step Functions 支援與 AWS CloudFormation. • 新增使用建立 Step Functions 的 Lambda 狀態機 AWS CloudFormation教學課程。	2017 年 2 月 10 日
更新	釐清 `ResultPath` 和 `OutputPath` 欄位與 `Parallel` 狀態相關的目前行為。	2017 年 2 月 6 日
更新	• 在教學中釐清狀態機器命名限制。 • 更正一些程式碼範例。	2017 年 1 月 5 日

變更	描述	變更日期
更新	更新 Lambda 函數範例以使用最新的程式設計模型。	2016 年 12 月 9 日
新功能	Step Functions 的初始版本。	2016 年 12 月 1 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

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。