aws

Guía para desarrolladores

AWS Step Functions

AWS Step Functions: Guía para desarrolladores

Table of Contents

¿Qué es AWS Step Functions?

AWS Step Functions es un servicio de flujo de trabajo visual que le ayuda a crear aplicaciones distribuidas, automatizar procesos, organizar microservicios y crear canalizaciones de datos y aprendizaje automático (ML).

En la consola gráfica de Step Functions, puede ver el flujo de trabajo de su aplicación como una serie de pasos basados en eventos.

Step Functions se basa en máquinas y tareas de estados. En Step Functions, las máquinas de estados se denominan flujos de trabajo, que son una serie de pasos basados en eventos. Cada paso de un flujo de trabajo se denomina estado. Por ejemplo, el estado de una tarea representa una unidad de trabajo que realiza otro AWS servicio, como llamar a otro Servicio de AWS o a una API.

Con los controles integrados de Step Functions, puede examinar el estado de cada paso de su flujo de trabajo para asegurarse de que la aplicación se ejecute en orden y según lo esperado. Según su caso de uso, puede hacer que Step Functions llame a AWS servicios, como Lambda, para realizar tareas. Puede crear flujos de trabajo que procesen y publiquen modelos de machine learning. Puede disponer de AWS servicios de control de Step Functions, por ejemplo AWS Glue, para crear flujos de trabajo de extracción, transformación y carga (ETL). También puede crear flujos de trabajo automatizados y de larga duración para aplicaciones que requieren la interacción humana.

> **Tip**
>
> Para aprender a usar Step Functions, siga los módulos interactivos del AWS Step Functions taller o lea la sección Primeros pasos de esta guía para crear un flujo de trabajo de solicitud de tarjetas de crédito.

Temas

- ¿Es la primera vez que utiliza Step Functions?

AWS SDK e integraciones optimizadas

Para llamar a otros AWS servicios, puedes usar las integraciones del AWS SDK de Step Functions o puedes usar una de las integraciones optimizadas de Step Functions.

- Las integraciones del AWS SDK te permiten llamar a cualquiera de los más de doscientos AWS servicios directamente desde tu máquina de estado, lo que te da acceso a más de nueve mil acciones de la API.
- Las integraciones optimizadas de Step Functions se han personalizado para simplificar el uso en sus máquinas de estado.

Flujos de trabajo estándar y rápidos

Step Functions tiene dos tipos de flujo de trabajo. Los flujos de trabajo estándar se ejecutan exactamente una vez y pueden durar hasta un año. Esto significa que cada paso de un flujo de trabajo estándar se ejecutará exactamente una vez. Sin embargo, los flujos de trabajo express tienen una ejecución at-least-once de flujo de trabajo y pueden ejecutarse durante un máximo de cinco minutos. Esto significa que uno o más pasos de un flujo de trabajo rápidos pueden ejecutarse más de una vez, mientras que cada paso del flujo de trabajo se ejecuta al menos una vez.

Las ejecuciones son instancias en las que se ejecuta el flujo de trabajo para realizar tareas. Los flujos de trabajo estándar son ideales para flujos de trabajo auditables y de larga duración, ya que muestran el historial de ejecución y la depuración visual. Los flujos de trabajo exprés son ideales para high-event-rate cargas de trabajo, como el procesamiento de datos en streaming y la ingesta de datos de IoT.

Especificaciones de flujos de trabajo estándar

- Índice de ejecución de 2000 por segundo
- Índice de transición de estado de 4000 por segundo
- Con precio por transición de estado
- Muestra el historial de ejecución y la depuración visual
- Admite todos los patrones y las integraciones de servicios

Especificaciones de flujos de trabajo rápidos

- Índice de ejecución de 100 000 por segundo
- Índice de transición de estado casi ilimitada
- Precio por número y duración de las ejecuciones
- Enviar el historial de ejecuciones a Amazon CloudWatch
- Muestra el historial de ejecución y la depuración visual en función del nivel de registro habilitado
- Admite todas las integraciones de servicios y la mayoría de patrones

Para obtener más información sobre flujos de trabajo estándar y rápidos, incluidos los precios de Step Functions, consulte los siguientes temas:

- Flujos de trabajo estándar en comparación con flujos de trabajo rápidos
- Precios de AWS Step Functions

Casos de uso

Step Functions gestiona los componentes y la lógica de la aplicación para que pueda escribir menos código y centrarse en crear y actualizar la aplicación rápidamente. En esta sección se describen los casos de uso habituales para trabajar con Step Functions.

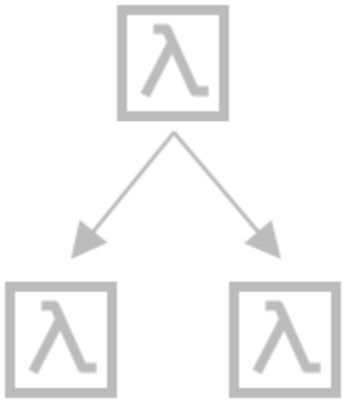
Caso de uso n.º 1: Orquestación de funciones

Cree un flujo de trabajo que ejecute un grupo de funciones de Lambda (pasos) en un orden específico. La salida de una función de Lambda pasa a la entrada de la siguiente función de Lambda. El último paso del flujo de trabajo arroja un resultado. Con Step Functions, puede ver cómo cada paso de su flujo de trabajo interactúa entre sí, de modo que puede asegurarse de que cada paso cumple su función prevista.

Para ver un tutorial que muestra cómo crear una máquina de estado con un grupo de funciones, consulte los siguientes temas:

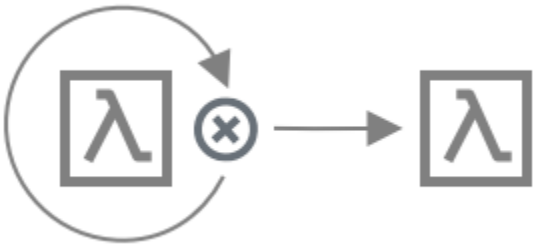
- Empezar con AWS Step Functions

Caso de uso n.º 2: ramificación



Un cliente solicita un aumento del límite de crédito. Al usar un estado `Choice`, puede hacer que Step Functions tome decisiones en función de las aportaciones del estado `Choice`. Si la solicitud supera el límite de crédito previamente aprobado por su cliente, puede hacer que Step Functions envíe la solicitud de su cliente a un administrador para que la apruebe. Si la solicitud es inferior al límite de crédito preaprobado por su cliente, puede hacer que Step Functions apruebe la solicitud automáticamente.

Caso de uso nº. 3: Gestión de errores



`Retry`

En este caso de uso, un cliente solicita un nombre de usuario. La primera vez, la solicitud de su cliente no es válida. Mediante una instrucción `Retry`, puede hacer que Step Functions vuelva a intentar la solicitud de su cliente. La primera vez, la solicitud de su cliente es válida.

`Catch`

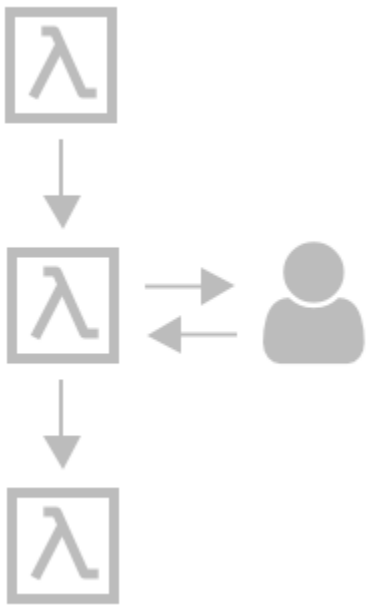
En un caso de uso similar, un cliente solicita un nombre de usuario no disponible. Mediante una instrucción `Catch`, Step Functions le sugiere un nombre de usuario disponible. Si su cliente utiliza

el nombre de usuario disponible, puede hacer que Step Functions pase al siguiente paso de su flujo de trabajo, que consiste en enviar un correo electrónico de confirmación. Si su cliente no utiliza el nombre de usuario disponible, haga que Step Functions vaya a otro paso de su flujo de trabajo, que consiste en volver a iniciar el proceso de registro.

Para ver ejemplos más detallados de instrucciones `Retry` y `Catch`, consulte lo siguiente:

- Control de errores en Step Functions

Caso de uso n.º 4: Humano en el bucle

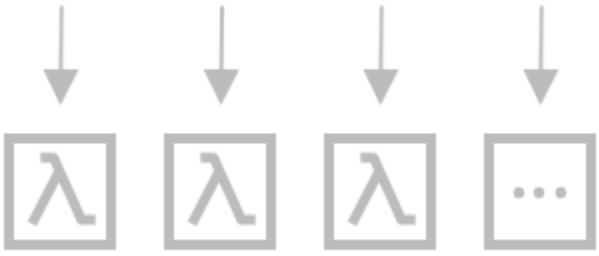


Con una aplicación bancaria, uno de tus clientes envía dinero a un amigo. El cliente espera un correo electrónico de confirmación. Con una devolución de llamada y un token de tarea, Step Functions le dice a Lambda que envíe el dinero a su cliente e informe cuando el amigo de su cliente lo reciba. Cuando Lambda informa de que el amigo de su cliente ha recibido el dinero, puede hacer que Step Functions pase al siguiente paso de su flujo de trabajo, que consiste en enviar a su cliente un correo electrónico de confirmación.

Para ver un ejemplo de proyecto en el que se muestra una devolución de llamada con un token de tarea, consulte lo siguiente:

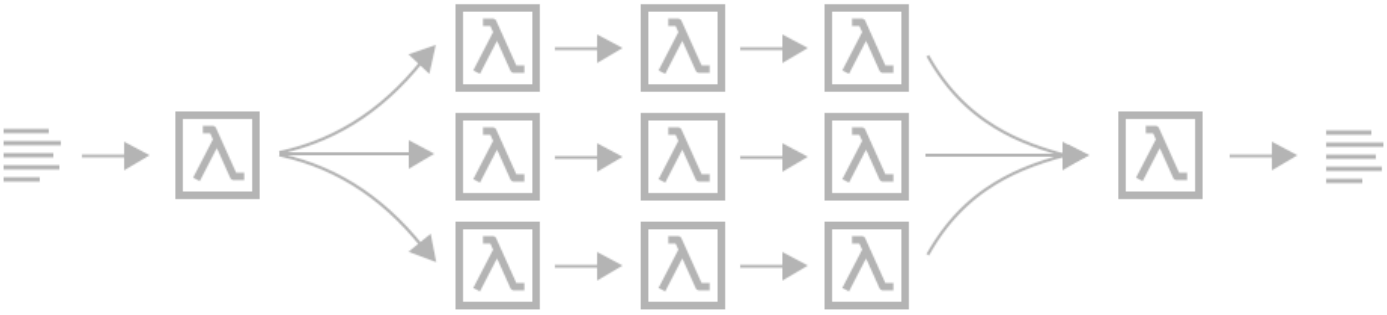
- Ejemplo de patrón de devolución de llamadas (Amazon SQS, Amazon SNS, Lambda)

Caso de uso n.º 5: procesamiento paralelo



Un cliente convierte un archivo de vídeo en cinco resoluciones de pantalla diferentes para que los espectadores puedan ver el vídeo en varios dispositivos. Mediante un estado `Parallel`, Step Functions introduce el archivo de vídeo para que Lambda pueda procesarlo en las cinco resoluciones de pantalla al mismo tiempo.

Caso de uso n.º 6: paralelismo dinámico



Un cliente pide tres artículos y debe preparar cada uno de ellos para su entrega. Comprueba la disponibilidad de cada artículo, reúne cada artículo y, a continuación, empaqueta cada artículo para su entrega. Mediante un estado `Map`, Step Functions hace que Lambda procese en paralelo cada uno de los artículos del cliente. Una vez que todos los artículos de su cliente estén empaquetados para su entrega, Step Functions pasa al siguiente paso de su flujo de trabajo, que consiste en enviar al cliente un correo electrónico de confirmación con información de seguimiento.

Para ver un proyecto de ejemplo que muestra el paralelismo dinámico mediante un estado `Map`, consulte lo siguiente:

- Procesamiento dinámico de datos con un estado Map

Integraciones de servicios

Step Functions se integra con varios AWS servicios. Para combinar Step Functions con estos servicios, utilice los siguientes patrones de integración de servicios:

Solicitar una respuesta (predeterminado)

- Llame a un servicio y deje que Step Funciones avance al siguiente estado después de que obtenga una respuesta HTTP.

Ejecutar un trabajo (.sync)

- Llame a un servicio y haga que Step Functions espere a que finalice un trabajo.

Espera a que te devuelvan la llamada con un token de tarea (. waitForTaskSímbolo)

- Llame a un servicio con un token de tarea y haga que Step Functions espere hasta que el token de tarea regrese con una devolución de llamada.

En la siguiente tabla se muestran las integraciones de servicios y los patrones de integración de servicios disponibles para Step Functions.

Los flujos de trabajo estándar y los flujos de trabajo exprés admiten las mismas integraciones pero no los mismos patrones de integración.

- La compatibilidad con los patrones de integración es diferente para cada integración.
- Los flujos de trabajo de Express no admiten Run a Job (.sync) ni Wait for Callback (). waitForTaskSímbolo).
- Para obtener más información, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

Standard Workflows

Integraciones de servicios compatibles

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
Integraciones optimizadas	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	✓		✓

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**.sync**)	Espere la devolución de la llamada (**.waitForTaskToken**)
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integraciones de SDK	Más de doscientas	✓		✓

Express Workflows

Integraciones de servicios compatibles

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**.sync**)	Espere la devolución de la llamada (**.waitForTaskToken**)
Integraciones	Amazon API Gateway	✓		
	Amazon Athena	✓		

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
optimizadas	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	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
	AWS Step Functions	✓		
AWS Integraciones de SDK	Más de doscientas	✓		

Regiones de admitidas

La mayoría de AWS las regiones admiten Step Functions. Para obtener una lista completa de AWS las regiones en las que Step Functions está disponible, consulta la tabla de AWS regiones.

¿Es la primera vez que utiliza Step Functions?

Si es la primera vez que utiliza Step Functions, los temas siguientes le ayudarán a comprender las distintas partes del trabajo con Step Functions, incluida la forma en que Step Functions se combina con otros AWS servicios:

- Tutoriales de Step Functions
- Proyectos de muestra para Step Functions
- AWS Step Functions SDK de ciencia de datos para Python

Empezar con AWS Step Functions

Step Functions es un servicio de orquestación sin servidor que permite definir el flujo de trabajo de una aplicación como una serie de pasos basados en eventos. Cada paso del flujo de trabajo se denomina estado. Lo más habitual es utilizar estados, como Estado de la tarea, Choice, Parallel y Map, para definir sus flujos de trabajo. Dentro de `Task` los estados, puede usar las integraciones de AWS SDK compatibles con Step Functions y organizar múltiples componentes de sus flujos de Servicios de AWS trabajo.

Temas

Conceptos clave

Antes de comenzar los tutoriales, revise los siguientes términos clave de Step Functions para ver el contexto.

Plazo	Descripción
Flujo de trabajo	Secuencia de pasos que a menudo reflejan un proceso empresarial.

Plazo	Descripción
Estados	Pasos individuales de su máquina de estados que pueden tomar decisiones en función de sus entradas, realizar acciones a partir de esas entradas y pasar las salidas a otros estados. Para obtener más información, consulte Estados.
Workflow Studio	Diseñador visual de flujos de trabajo que ayuda a crear prototipos y flujos de trabajo con mayor rapidez. Para obtener más información, consulte AWS Step Functions Estudio de flujo de trabajo.
Máquina de estado	Flujo de trabajo definido mediante texto JSON que representa los estados o pasos individuales del flujo de trabajo junto con campos como `StartAt`, `TimeoutSeconds` y `Version`. Para obtener más información, consulte Estructura de las máquinas de estado.
Amazon States Language	Lenguaje estructurado y basado en JSON que se utiliza para definir las máquinas de estados. Con el ASL, puede definir un conjunto de estados que pueden funcionar (`Task`estado), determinar qué estados pasar a los siguientes (`Choice`estado) y detener una ejecución con un error (`Fail`estado). Para obtener más información, consulte Lenguaje de estados de Amazon.
Configuración de entrada y salida	Los estados de un flujo de trabajo reciben datos JSON como entrada y, por lo general, pasan los datos JSON como salida al siguiente estado. Step Functions proporciona filtros para controlar el flujo de datos entre estados. Para obtener más información, consulte Procesamiento de entrada y salida en Step Functions.
Integración con los servicios	Puedes llamar a las acciones de la API del AWS servicio desde tu flujo de trabajo. Para obtener más información, consulte Uso AWS Step Functions con otros servicios.

Plazo	Descripción
Tipo de integración de servicios	• AWS Integraciones de SDK: forma estándar de llamar a cualquiera de las más de doscientas Servicios de AWS y más de nueve mil acciones de la API directamente desde tu máquina de estados. • Integraciones optimizadas: integraciones personalizadas que agilizan las llamadas y el intercambio de datos con determinados servicios. Por ejemplo, Lambda Invoke convertirá automáticamente el `Payload` campo de la respuesta de una cadena JSON de escape en un objeto JSON.
Patrón de integración de servicios	Al llamar a un Servicio de AWS, se utiliza uno de los siguientes patrones de integración de servicios: • Solicitar una respuesta (predeterminado): llama a un servicio y pasa al siguiente estado inmediatamente después de recibir una respuesta HTTP. • Ejecutar un trabajo (.sync): llama a un servicio y hace que Step Functions espere a que se complete un trabajo. • Espera a que te devuelvan la llamada con un token de tarea (.wait ForTask Token): llama a un servicio con un token de tarea y haz que Step Functions espere hasta que el token de la tarea regrese con una devolución de llamada.
Ejecución	Las ejecuciones de máquinas de estado son instancias en las que se ejecuta un flujo de trabajo para realizar tareas. Para obtener más información, consulte Ejecuciones en Step Functions.

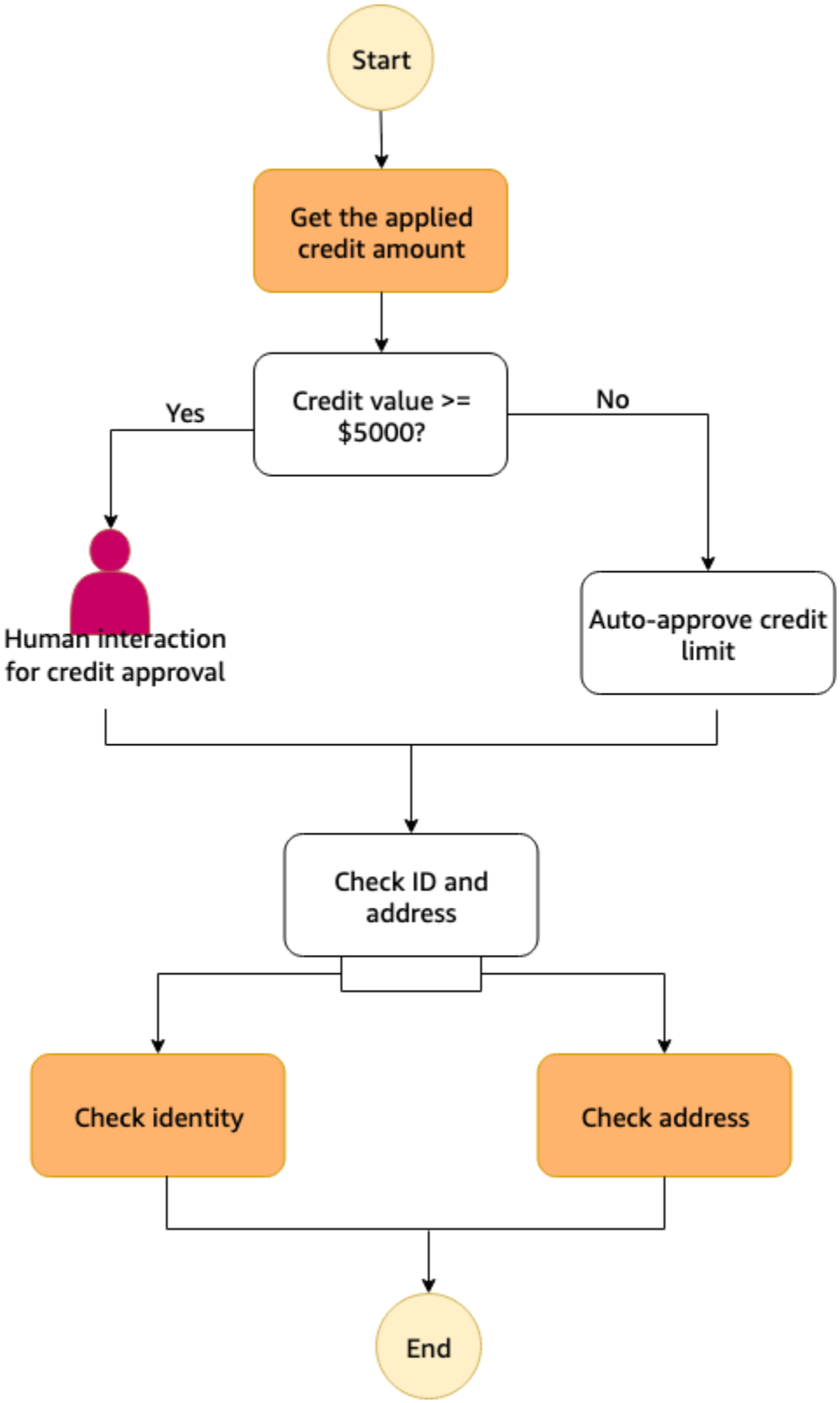
Tutoriales de esta serie

Tras completar estos tutoriales, dispondrás de un flujo de trabajo que simula el procesamiento de una solicitud de tarjeta de crédito. Aprenderás a usar estados comunes e integrar tu flujo de trabajo con otros Servicios de AWS

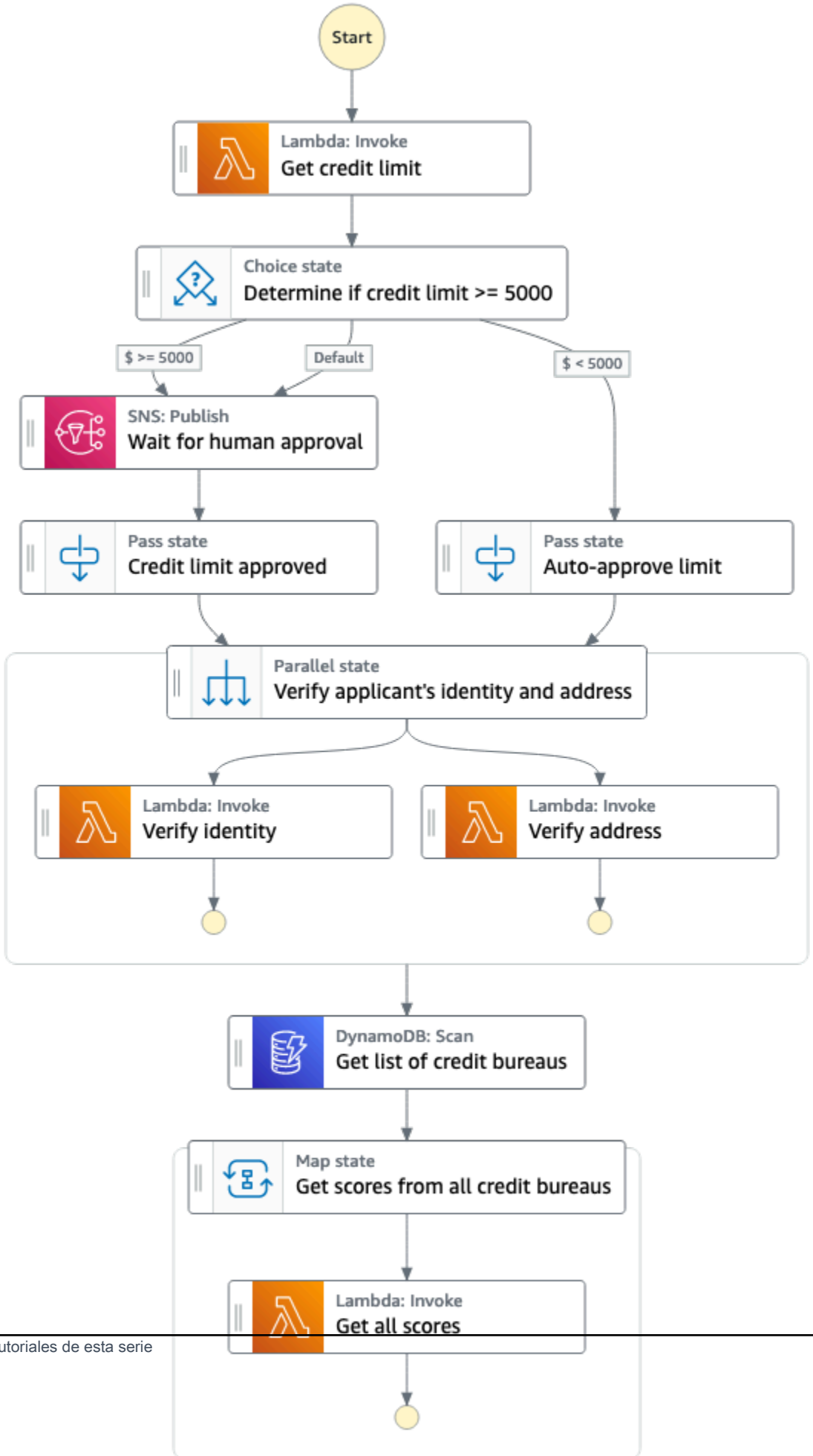
Step Functions se puede utilizar para crear muchos tipos de flujos de trabajo, como el procesamiento de datos, la automatización de TI, el aprendizaje automático y la codificación multimedia.

El siguiente diagrama de flujo describe los pasos que debe seguir una empresa para procesar una solicitud de tarjeta de crédito. Si el importe del crédito solicitado es inferior a 5000$, el límite de

crédito se aprobará automáticamente. Si la solicitud supera el límite, el flujo de trabajo incluirá a una persona encargada de verificar la identidad de los solicitantes y revisar las calificaciones crediticias.

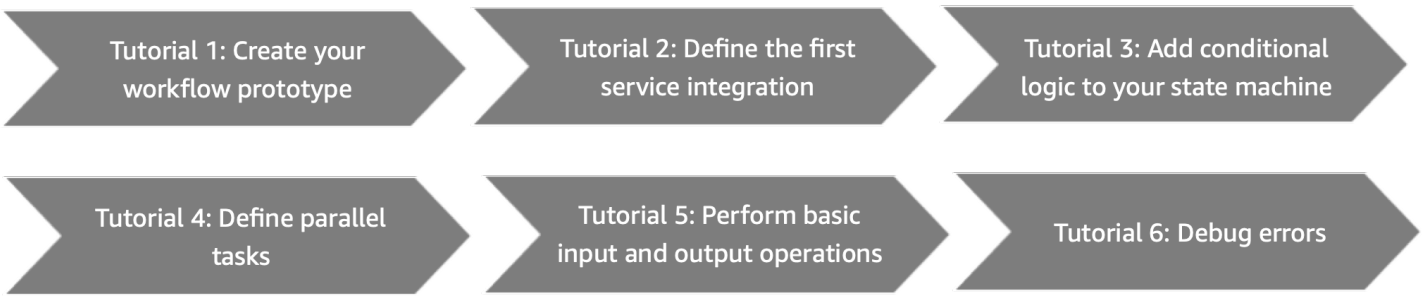


El siguiente diagrama muestra cómo los pasos del proceso empresarial de la solicitud de crédito se representan mediante estados en un flujo de trabajo de Step Functions.


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

En la siguiente serie de tutoriales, creará el flujo de trabajo de procesamiento de tarjetas de crédito.

Recomendamos completar estos tutoriales para aprender las funciones clave de Step Functions.



Antes de comenzar, asegúrese de completar los requisitos previos.

Requisitos previos para empezar con AWS Step Functions

Antes de usarlo AWS Step Functions por primera vez, complete las siguientes tareas.

Inscríbase en una Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en https://aws.amazon.com/ y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión AWS Management Consolecomo propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

   Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In .
2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .
2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la Guía del AWS IAM Identity Center usuario.

Iniciar sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, utilice la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte Iniciar sesión en el portal de AWS acceso en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .
2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

   Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center .

Tutorial 1: Crear el prototipo para la máquina de estado

En este tutorial creará el prototipo para el flujo de trabajo de procesamiento de tarjetas de crédito utilizando Workflow Studio de Step Functions. Elegirá las acciones y los estados de la API necesarios en las pestañas Acciones y Flujo, respectivamente, y utilizará la característica de arrastrar y soltar de Workflow Studio para crear el prototipo de flujo de trabajo. En los siguientes tutoriales aprenderá a configurar los Servicios de AWS y estados de Step Functions que utilizará en este flujo de trabajo.

Para crear el prototipo de la máquina de estado

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. En la pestaña Acciones de Workflow Studio, arrastre una acción de la API Invocar AWS Lambda y suéltala en el estado vacío con la etiqueta Arrastrar la primera acción aquí. Establezca la siguiente configuración:

   - En la pestaña Configuración, en Nombre de estado, escriba **`Get credit limit`**.
5. En la pestaña Flujo, arrastre y suelte un estado Elección debajo del estado Obtener límite de crédito. Cambie el nombre del estado Elección a **`Credit applied >= 5000?`**.
6. Arrastre y suelte los siguientes estados como ramas del estado Crédito aplicado >= 5000?.

   a. Publicar Amazon SNS: en la pestaña Acciones, arrastre y suelte la acción de la API Publicar Amazon SNS. Cambie el nombre de este estado a **`Wait for human approval`**.
   b. Estado Aprobado: en la pestaña Flujo, arrastre y suelte el estado Aprobado. Cambie el nombre de esta rama a **`Auto-approve limit`**.
7. Arrastre y suelte un estado Aprobado debajo del estado Esperar aprobación humana. Cambie el nombre de este estado Aprobado a **`Credit limit approved`**.
8. Arrastre y suelte un estado En paralelo detrás del estado Elección de la siguiente manera:
   a. Arrastre el estado En paralelo detrás del estado Límite de crédito aprobado.
   b. Cambie el nombre del estado En paralelo a **`Verify applicant's identity and address`**.
   c. En las dos ramas del estado En paralelo, arrastre y suelte dos acciones de la API Invocar AWS Lambda.
   d. Cambie el nombre de estos estados a **`Verify identity`** y **`Verify address`**, respectivamente.
   e. Elija el estado Aprobación automática de límite y, en Siguiente estado, seleccione Verificar identidad y dirección del solicitante.
9. Arrastre un estado DynamoDB Scan y colóquelo debajo del estado Verificar identidad y dirección del solicitante. Cambie el nombre del estado DynamoDB Scan a **`Get list of credit bureaus`**.
10. Arrastre y suelte un estado Mapa detrás del estado Obtener lista de agencias de crédito. Configure el estado Mapa de este modo:
    a. Cambie su nombre a **`Get scores from all credit bureaus`**.
    b. En Modo de procesamiento, mantenga la selección predeterminada de Inline.
    c. Arrastre y suelte una acción de la API Invocar AWS Lambda en el estado vacío con la etiqueta Colocar estado aquí.
    d. Cambie el nombre del estado Invocar AWS Lambda a **`Get all scores`**.
11. Mantenga esta ventana abierta y continúe con el siguiente tutorial para realizar más acciones.

Pasos siguientes

En el siguiente tutorial, aprenderá a integrar la función de Lambda utilizada por el estado Obtener límite de crédito.

Tutorial 2: Definir la primera integración de servicios mediante una función de Lambda

En este tutorial aprenderá a definir la primera integración de servicios para su flujo de trabajo. Se utiliza el estado de `Task` llamado Obtener límite de crédito para invocar una función de Lambda. Dentro de `Task` los estados, puedes usar las integraciones de AWS SDK compatibles con Step Functions.

Para definir la primera integración de servicios para su flujo de trabajo, cree una función de Lambda. A continuación, actualice el flujo de trabajo para especificar la integración de servicios con la función de Lambda. La función de Lambda utilizada en este tutorial devuelve un entero generado aleatoriamente que representa el límite de crédito que ha pedido un solicitante.

Temas

Paso 1: Crear y probar la función de Lambda.

Puedes escribir código para la función en el editor AWS Management Console o en tu editor favorito. En los pasos siguientes, creará una función de Lambda Node.js con el título `RandomNumberforCredit`.

> **Important**
>
> Asegúrese de que el prototipo de flujo de trabajo que creó en el tutorial 1 sea Región de AWS igual que la función Lambda que creará en este tutorial.

1. En una nueva pestaña o ventana, abra la consola Lambda y cree una función de Lambda Node.js 16.x con el título **`RandomNumberforCredit`**. Para obtener información sobre cómo crear una función de Lambda utilizando la consola, consulte Cree una función de Lambda con la consola en la Guía para desarrolladores de AWS Lambda .
2. En la RandomNumberforCreditpágina, elija index.mjs y sustituya el código existente en el área Código fuente por el siguiente código.

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

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

};
```

3. En la sección Información general de la función, copie el nombre del recurso de Amazon de la función de Lambda y guárdelo en un archivo de texto. Necesitará el ARN de la función al especificar la integración de servicios para el estado Obtener límite de crédito. A continuación se muestra un ejemplo de ARN:

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

4. Elija Implementar y, a continuación, elija Probar para implementar los cambios y ver el resultado de la función de Lambda.

Paso 2: Actualizar el flujo de trabajo: configurar el estado de Obtener límite de crédito

En la consola de Step Functions, actualizará el flujo de trabajo para especificar la integración de servicios con la función de Lambda que creó en el paso 1 `RandomNumberforCredit`.

1. Abra la ventana de la consola de Step Functions que contiene el prototipo de flujo de trabajo que creó en el Tutorial 1.
2. Elija el estado Obtener límite de crédito y haga lo siguiente en la pestaña Configuración:
   a. Para Tipo de integración, mantenga la selección predeterminada de Optimizado.

      Con Step Functions, puede integrarse con otras Servicios de AWS y organizarlas en sus flujos de trabajo. Para obtener más información sobre las integraciones de servicios y sus tipos, consulte Uso AWS Step Functions con otros servicios.

   b. En Nombre de función, elija la función RandomNumberforCreditLambda en la lista desplegable.
   c. Mantenga las selecciones predeterminadas para el resto de los elementos.
3. Mantenga esta ventana abierta y continúe con el siguiente tutorial para realizar más acciones.

> **Note**
>
> En este tutorial, aprendió a realizar la integración con una función de Lambda dentro de un estado de `Task` en sus flujos de trabajo. También puedes usar otras integraciones de AWS SDK compatibles en el `Task` estado especificando el nombre del servicio y la llamada a la API, como se muestra en la siguiente sintaxis:
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> Para obtener más información, consulte Uso AWS Step Functions con otros servicios.

Siguientes pasos

En el siguiente tutorial, implementará lógica condicional en el flujo de trabajo. La lógica condicional de las máquinas de estado de Step Functions se comporta de forma similar a una instrucción if-else en la mayoría de los lenguajes de programación comunes. Utilizará lógica condicional en el flujo de trabajo para determinar la ruta de ejecución basándose en información condicional.

Tutorial 3: Implementar una condición if-else en el flujo de trabajo

Puede implementar condiciones if-else en sus flujos de trabajo utilizando el estado `Choice`. Determina la ruta de ejecución del flujo de trabajo en función de si una condición específica se evalúa como verdadera o falsa.

En este tutorial, añadirá una lógica condicional para determinar si el importe de crédito aplicado devuelto por la función de Lambda `RandomNumberforCredit` utilizada en el Tutorial 2 supera un límite del umbral específico. Si el importe supera el límite del umbral, la solicitud requiere una interacción humana para su aprobación. De lo contrario, la solicitud se aprueba automáticamente y continúa con el siguiente paso.

Imitará el paso de la interacción humana pausando la ejecución del flujo de trabajo hasta que se devuelva un token de tarea. Para ello, pasará un token de tarea a la integración del SDK AWS que utilizará en este tutorial, que es Amazon Simple Notification Service. La ejecución del flujo de trabajo se detendrá hasta que reciba el token de la tarea mediante una llamada a la API `SendTaskSuccess`. Para obtener más información sobre el uso de tokens de tareas, consulte Cómo esperar una devolución de llamada con el token de tarea.

Como ya ha definido los pasos para la aprobación humana y la aprobación automática en su prototipo de flujo de trabajo, en este tutorial, primero debe crear un tema de Amazon SNS que reciba el token de devolución de llamada. A continuación, se crea una función de Lambda para implementar la funcionalidad de devolución de llamada. Por último, actualiza el prototipo de flujo de trabajo añadiendo los detalles de estas integraciones Servicio de AWS.

Temas

Paso 1: Crear un tema de Amazon SNS que reciba el token de devolución de llamada

Para implementar el paso de interacción humana, debe publicarlo en un tema del Amazon Simple Notification Service y pasar el token de la tarea de devolución de llamada a este tema. La tarea de devolución de llamada pausará la ejecución del flujo de trabajo hasta que se devuelva el token de la tarea con una carga útil.

1. Abra la consola de Amazon SNS y cree un tipo de tema Standard. Para obtener más información acerca de la creación de un tema, consulte Creación de un tema de Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.
2. Especifique el nombre del tema como **`TaskTokenTopic`**.
3. Asegúrese de copiar el ARN del tema y guardarlo en un archivo de texto. Necesitará el tema ARN al especificar la integración del servicio para el estado Wait for human approval. A continuación se muestra un ejemplo de ARN:

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

4. Cree una suscripción por correo electrónico para el tema y, a continuación, confirme su suscripción. Para obtener información sobre la suscripción a un tema, consulte Crear una suscripción al tema en la Guía para desarrolladores de Amazon Simple Notification Service.

Paso 2: Crear una función de Lambda para gestionar la devolución de llamada

Para gestionar la funcionalidad de devolución de llamadas, definirá una función de Lambda y añadirá el tema de Amazon SNS que creó en el paso 1 como activador de esta función. Cuando publica en el tema de Amazon SNS con un token de tarea, la función de Lambda se invoca con la carga del mensaje publicado.

- Paso 2.1: Crear una función de Lambda para gestionar la devolución de llamada
- Paso 2.2: Añadir el tema Amazon SNS como activador de la función de Lambda
- Paso 2.3: Proporcionar los permisos necesarios para el rol de IAM de la función de Lambda

Paso 2.1: Crear una función de Lambda para gestionar la devolución de llamada

En esta función, procesará la solicitud de aprobación del límite de crédito y devolverá el resultado de la solicitud como si se hubiera realizado correctamente con la llamada a la API `SendTaskSuccess`. Esta función de Lambda también devolverá el token de tarea que recibió del tema Amazon SNS.

Para simplificar, la función de Lambda utilizada para el paso de interacción humana aprueba automáticamente cualquier tarea y devuelve el token de la tarea con una llamada a la API `SendTaskSuccess`. Puede denominar la función de Lambda como **`callback-human-approval`**.

1. En una nueva pestaña o ventana, abra la consola Lambda y cree una función de Lambda Node.js 16.x con el título **`callback-human-approval`**. Para obtener información sobre cómo crear una función de Lambda utilizando la consola, consulte Cree una función de Lambda con la consola en la Guía para desarrolladores de AWS Lambda.
2. En la página callback-human-approval, sustituya el código existente en el área Código fuente por el código siguiente.

```
// 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. Mantenga esta ventana abierta y ejecute los pasos de la siguiente sección para realizar más acciones.

Paso 2.2: Añadir el tema Amazon SNS como activador de la función de Lambda

Al añadir el tema de Amazon SNS que creó en el paso 1 de este tutorial como activador de la función de Lambda que creó en el paso 2.1 de este tutorial, la función de Lambda se activa cada vez que publica en el tema Amazon SNS. Cuando publica en el tema de Amazon SNS con un token de tarea, la función de Lambda se invoca con la carga del mensaje publicado. Para obtener más información sobre la configuración de desencadenadores para funciones de Lambda, consulte Configuración de desencadenadores en la Guía para desarrolladores de AWS Lambda.

1. En la sección Información general de la función de Lambda `callback-human-approval`, elija Añadir desencadenador.
2. En la lista desplegable de desencadenadores, elija SNS como el desencadenador.
3. Para Tema de SNS, comience a escribir el nombre del tema de Amazon SNS que creó en el paso 1 de este tutorial y selecciónelo en la lista desplegable que aparece.
4. Elija Añadir.

5. Mantenga esta ventana abierta y ejecute los pasos de la siguiente sección para realizar más acciones.

Paso 2.3: Proporcionar los permisos necesarios para el rol de IAM de la función de Lambda

Debe proporcionar a la función de Lambda `callback-human-approval` los permisos de acceso a Step Functions para devolver el token de la tarea junto con la llamada a la API `SendTaskSucess`.

1. En la página callback-human-approval, seleccione la pestaña Configuración y, a continuación, elija Permisos.
2. En Rol de ejecución, elija el nombre del rol para acceder a la página Roles de la consola de AWS Identity and Access Management.
3. Para añadir el permiso necesario, seleccione Añadir permisos y, a continuación, seleccione Asociar políticas.
4. En el cuadro de búsqueda, escriba **`AWSStepFunctions`** y, a continuación, pulse Entrar.
5. Elija AWSStepFunctionsFullAccess y, a continuación, desplácese hacia abajo para seleccionar Asociar políticas. Esto añade la política que contiene el permiso necesario para el rol de la función de Lambda `callback-human-approval`.

Paso 3: Actualizar el flujo de trabajo: añadirá la lógica de condición if-else en el estado Choice

En la consola de Step Functions, defina la lógica condicional para su flujo de trabajo mediante el estado `Choice`. Si el resultado devuelto por la función de Lambda `RandomNumberforCredit` es inferior a 5000, el crédito solicitado se aprueba automáticamente. Si el resultado devuelto es superior o igual a 5000, la ejecución del flujo de trabajo pasa al paso de interacción humana para la aprobación del límite de crédito.

En el estado `Choice`, se utiliza un operador de comparación para comparar una variable de entrada con un valor específico. Puede especificar la variable de entrada como entrada de ejecución al iniciar una ejecución de máquina de estado o utilizar la salida de un paso anterior como entrada para el paso actual. De forma predeterminada, la salida de un paso se almacena en una variable llamada `Payload`. Para usar el valor de la variable `Payload` para comparar en el estado `Choice`, utilice la sintaxis $ que se muestra en el siguiente procedimiento.

Para obtener información sobre cómo fluye la información de un estado a otro y sobre cómo especificar la entrada y la salida en los flujos de trabajo, consulte Tutorial 7: Configurar entrada y salida y Procesamiento de entrada y salida en Step Functions.

> **Note**
>
> Si el estado `Choice` utiliza una variable de entrada especificada en la entrada de ejecución de la máquina de estado para la comparación, utilice la sintaxis `$.variable_name` para realizar la comparación. Por ejemplo, para comparar una variable, por ejemplo `myAge`, utilice la sintaxis`$.myAge`.

Como en este paso, el estado `Choice` recibirá la entrada del estado Obtener límite de crédito, utilizará la sintaxis `$` para la configuración del estado `Choice`. Para ver en qué se diferencia el resultado de la ejecución de la máquina de estado cuando se utiliza la sintaxis `$.variable_name` de la configuración de estado `Choice` para hacer referencia al resultado de un paso anterior, consulte la sección Depuración de la ruta no válida: error de elección de estado del Tutorial 8.

Para añadir la lógica de la condición if-else mediante el estado **`Choice`**

1. Abra la ventana de la consola de Step Functions que contiene el prototipo de flujo de trabajo que creó en Tutorial 1: Crear el prototipo para la máquina de estado.
2. Elija el estado Credit applied >= 5000? y en la pestaña Configuración, especifique la lógica condicional de la siguiente manera:
   a. En Reglas de elección, elija el icono de edición en el mosaico Regla n.º 1 para definir la primera regla de elección.
   b. Elija Añadir condiciones.
   c. En el cuadro de diálogo Condiciones de la regla n.º 1, para Variable, introduzca **`$`**.
   d. En Operador, elija es menor que.
   e. En Valor, elija Número constante y, a continuación, introduzca **`5000`** en el campo situado junto a la lista desplegable Valor.
   f. Seleccione Guardar condiciones.
   g. En la lista desplegable El siguiente estado es:, seleccione Aprobar automáticamente el límite.

h. Seleccione Añadir nueva regla de selección y, a continuación, defina la segunda regla cuando el importe del crédito sea superior o igual a 5000 repitiendo los subpasos 2.b a 2.f. En Operador, seleccione es mayor o igual a.

i. En la lista desplegable El siguiente estado es:, seleccione Esperar la aprobación humana.

j. En el cuadro Regla predeterminada, elija el icono de edición para definir la regla de elección predeterminada y, a continuación, elija Esperar la aprobación humana en la lista desplegable Estado predeterminado. Defina la regla predeterminada para especificar el siguiente estado al que realizar la transición si ninguna de las condiciones del estado de Elección resulta verdadera o falsa.

3. Configure el estado Wait for human approval de la siguiente manera:

a. En la pestaña Configuración, en Tema, comience a escribir el nombre del tema de Amazon SNS, TaskTokenTopic y elija el nombre tal como aparece en la lista desplegable.

b. En Mensaje, seleccione Introducir mensaje en la lista desplegable. En el campo Mensaje, especifique el mensaje que desea publicar en el tema Amazon SNS. Para este tutorial, debe publicar un token de tarea como mensaje.

   Un token de tarea permite pausar un flujo de trabajo de Step Functions de tipo Standard hasta que se complete un proceso externo y se devuelva el token de tarea. Al especificar un estado Task como tarea de devolución de llamada especificando el patrón de integración de servicios de `.waitForTaskToken`, se genera un token de tarea que se coloca en el objeto de contexto cuando se inicia la tarea. El objeto de contexto es una estructura JSON interna que está disponible durante una ejecución y contiene información sobre la máquina de estado y su ejecución. Para obtener más información acerca de la copia de objetos, consulte Objeto Context (Contexto).

c. En el cuadro que se muestra, escriba lo siguiente como mensaje:

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

d. Seleccione la casilla de verificación Esperar a que te devuelvan la llamada.

e. En el cuadro de diálogo que aparece, elija Listo.

4. Mantenga esta ventana abierta y continúe con el siguiente tutorial para realizar más acciones.

Pasos siguientes

En el siguiente tutorial, aprenderá a realizar varias tareas en paralelo.

Tutorial 4: Definir varias tareas para realizarlas en paralelo

Hasta ahora, ha aprendido a ejecutar flujos de trabajo de forma secuencial. Sin embargo, puede ejecutar dos o más pasos en paralelo utilizando el estado `Parallel`. Un estado `Parallel` hace que el intérprete ejecute cada ramificación simultáneamente.

Las dos ramificaciones de un estado `Parallel` reciben la misma entrada, pero cada ramificación procesa las partes de la entrada específicas para ella. Step Functions espera a que cada ramificación termine de ejecutarse antes de ir al paso siguiente.

En este tutorial, utilizará el estado Parallel para comprobar simultáneamente la identidad y la dirección del solicitante.

Temas

Paso 1: Crear las funciones de Lambda para realizar las comprobaciones necesarias

Este flujo de trabajo de solicitud de tarjetas de crédito invoca dos funciones de Lambda dentro del estado Parallel para comprobar la identidad y la dirección del solicitante. Estas comprobaciones se realizan simultáneamente mediante el estado Parallel. La máquina de estado completa la ejecución solo después de que ambas ramificaciones paralelas hayan completado la ejecución.

Para crear las funciones de Lambda de verificación de identidad y dirección de verificación

1. En una nueva pestaña o ventana, abra la consola de Lambda y cree una función de Lambda Node.js 16.x con el título `check-identity` y `check-address`. Para obtener información sobre cómo crear una función de Lambda utilizando la consola, consulte Cree una función de Lambda con la consola en la Guía para desarrolladores de AWS Lambda.
2. Abra la página de la función de check-identity y sustituya el código existente en el área Código fuente por el siguiente código:

```
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. Abra la página de la función de check-address y sustituya el código existente en el área Código fuente por el siguiente código:

```
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. Para ambas funciones de Lambda, desde la sección Descripción general de la función, copie sus respectivos nombres de recursos de Amazon (ARN) y guárdelos en un archivo de texto. Necesitará los ARN de la función para especificar la integración del servicio para el estado Verify applicant's identity and address. A continuación se muestra un ejemplo de ARN:

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

Paso 2: Actualizar el flujo de trabajo: añadir tareas paralelas a realizar

En la consola de Step Functions, actualizará el flujo de trabajo para especificar la integración de servicios con las funciones de Lambda check-identity y check-address creadas en el paso 1.

Para añadir tareas paralelas al flujo de trabajo

1. Abra la ventana de la consola de Step Functions que contiene el prototipo de flujo de trabajo que creó en Tutorial 1: Crear el prototipo para la máquina de estado.
2. Elija el estado Verify identity y haga lo siguiente en la pestaña Configuración:

a. Para Tipo de integración, mantenga la selección predeterminada de Optimizado.

> **Note**
>
> Con Step Functions, puede integrarse con otras Servicios de AWS y organizarlas en sus flujos de trabajo. Para obtener más información sobre las integraciones de servicios y sus tipos, consulte Uso AWS Step Functions con otros servicios.

b. En Nombre de la función, elija la función de Lambda check-identity en la lista desplegable.

c. En Carga, elija Introducir carga y, a continuación, sustituya el ejemplo de carga por lo siguiente como carga:

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

3. Elija el estado Verify address y haga lo siguiente en la pestaña Configuración:

a. Para Tipo de integración, mantenga la selección predeterminada de Optimizado.

b. En Nombre de la función, elija la función de Lambda check-address en la lista desplegable.

c. En Carga, elija Introducir carga y, a continuación, sustituya el ejemplo de carga por lo siguiente como carga:

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

4. Elija Siguiente.

Tutorial 5: Repetir simultáneamente en una colección de objetos

En el tutorial anterior, aprendió a ejecutar ramificaciones separadas de pasos en paralelo utilizando el estado `Parallel`. Utilice el estado `Map`, puede ejecutar un conjunto de pasos de flujo de trabajo

para cada elemento de un conjunto de datos. Las iteraciones del estado `Map` se ejecutan en paralelo, lo que permite procesar un conjunto de datos rápidamente.

Al incluir el estado `Map` en sus flujos de trabajo, puede realizar tareas, como el procesamiento de datos, utilizando uno de los dos Modos de procesamiento del estado Map: el modo En línea y el modo distribuido. Para configurar un estado `Map`, debe definir un estado `ItemProcessor`, que contiene objetos JSON que especifican el modo de procesamiento del estado `Map` y su definición. En este tutorial, ejecutará el estado `Map` en el modo En línea predeterminado, que admite hasta 40 iteraciones simultáneas. Cuando ejecuta el estado `Map` en modo Distribuido, admite hasta 10 000 ejecuciones paralelas de flujos de trabajo secundarios.

Cuando la ejecución del flujo de trabajo entre en ese estado `Map`, se repetirá sobre una matriz JSON especificada en la entrada de estado. Para cada elemento de la matriz, la iteración correspondiente se ejecuta en el contexto del flujo de trabajo que contiene el estado `Map`. Cuando se completen todas las iteraciones, el estado `Map` devolverá una matriz que contiene la salida de cada elemento procesado por el `ItemProcessor`.

En este tutorial, aprenderá a utilizar el estado `Map` en el modo En línea para obtener la calificación crediticia de un solicitante consultando un conjunto de agencias de crédito. Para ello, primero debe buscar los nombres de todas las agencias de crédito almacenadas en una tabla de Amazon DynamoDB y, a continuación, utilizar el estado `Map` para recorrer la lista de agencias de crédito y obtener la calificación crediticia del solicitante reportada por cada una de estas agencias.

Temas

- Paso 1: Crear una tabla de DynamoDB para almacenar el nombre de todas las agencias de crédito
- Paso 2: Actualizar la máquina de estado: obtener los resultados de la tabla de DynamoDB
- Paso 3: Crear una función de Lambda que devuelve las puntuaciones de crédito de todas las agencias de crédito
- Paso 4: Actualizar la máquina de estado: añadir un estado del mapa para obtener las puntuaciones crediticias de forma iterativa

Paso 1: Crear una tabla de DynamoDB para almacenar el nombre de todas las agencias de crédito

En este paso, se crea una tabla denominada **`GetCreditBureau`** mediante la consola de DynamoDB. La tabla utiliza el atributo de cadena Nombre como clave de Partición. En esta tabla,

guarda el nombre de todas las agencias de crédito de las que desea obtener la calificación crediticia del solicitante.

1. Inicie sesión en la AWS Management Console y abra la consola de DynamoDB en https://console.aws.amazon.com/dynamodb/.
2. En el panel de navegación de la consola, seleccione Tablas y, a continuación, seleccione Crear tabla.
3. Introduzca los datos siguientes para la tabla:
   a. En Nombre de la tabla, escriba **`GetCreditBureau`**.
   b. Escriba como clave de partición **`Name`**.
   c. Mantenga las selecciones predeterminadas y elija Crear tabla.
4. Una vez creada la tabla, en la lista Tablas, elija la tabla GetCreditBureau.
5. Elija Acciones y, a continuación, seleccione Crear elemento.
6. En Valor, introduzca el nombre de una agencia de crédito. Por ejemplo, **`CredTrack`**.
7. Elija Crear elemento.
8. Repita este proceso y cree elementos para los nombres de otras agencias de crédito. Por ejemplo, **`KapFinn`** y **`CapTrust`**.

Paso 2: Actualizar la máquina de estado: obtener los resultados de la tabla de DynamoDB

En la consola de Step Functions, añadirá un estado `Task` y utilizará la integración del SDK AWS para obtener los nombres de las agencias de crédito de la tabla de DynamoDB que creó en el paso 1. Utilizará el resultado de este paso como entrada para el estado `Map` que añadirá más adelante en el flujo de trabajo de este tutorial.

1. Abra la máquina de estado CreditCardWorkflow para actualizarla.
2. Seleccione el estado Get list of credit bureaus.
3. Para Parámetros de API, especifique el valor del Nombre de la tabla como **`GetCreditBureau`**.

Paso 3: Crear una función de Lambda que devuelve las puntuaciones de crédito de todas las agencias de crédito

En este paso, se crea una función de Lambda que recibe los nombres de todas las agencias de crédito como entrada y devuelve la calificación crediticia del solicitante para cada una de estas agencias de crédito. Esta función de Lambda se invocará desde el estado `Map` que añadirá al flujo de trabajo en el paso 4 de este tutorial.

1. Cree una función de Lambda Node.js 16.x y asígnele el nombre **`get-credit-score`**.
2. En la página titulada get-credit-score, pegue el siguiente código en el área Código fuente.

```
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. Implemente la función de Lambda.

Paso 4: Actualizar la máquina de estado: añadir un estado del mapa para obtener las puntuaciones crediticias de forma iterativa

En la consola de Step Functions, agregue un estado `Map` que invoca la función de Lambda get-credit-score para comprobar la calificación crediticia del solicitante de todas las agencias de crédito devueltas por el estado Get list of credit bureaus.

1. Abra la máquina de estado CreditCardWorkflow para actualizarla.
2. Elija el estado Get scores from all credit bureaus.

3. En la pestaña Configuración, elija Proporcionar una ruta a la matriz de elementos y, a continuación, introduzca **`$.Items`**.
4. Seleccione el paso Obtener todas las puntuaciones dentro del estado `Map`.
5. En la pestaña Configuración, asegúrese de seleccionar para Tipo de integración, Optimizado.
6. En nombre de Función, comience a escribir el nombre de la función de Lambda get-credit-score y selecciónela en la lista desplegable que aparece.
7. Para Carga, elija Ninguna carga.

Tutorial 6: Guardar el flujo de trabajo y ejecutar la máquina de estado

Ahora que has configurado los recursos de todos los que Servicios de AWS utilizas en el prototipo de flujo de trabajo, puedes guardarlo como una máquina de estados de Step Functions y empezar a ejecutarlo.

Temas

- Paso 1: Revisar la definición de máquina de estado generada automáticamente y guarde la máquina de estado
- Paso 2: Agregar las políticas de IAM restantes
- Paso 3: Ejecutar la máquina de estado

Paso 1: Revisar la definición de máquina de estado generada automáticamente y guarde la máquina de estado

A medida que arrastra y suelta estados de la pestaña Flow en el lienzo de Workflow Studio para crear el prototipo de flujo de trabajo, Step Functions compone automáticamente la definición de Lenguaje de estados de Amazon (ASL) del flujo de trabajo en tiempo real. Puede editar esta definición según sea necesario en el Editor de código.

Para revisar la definición de ASL y guardar la máquina de estado

1. (Opcional) Seleccione Definición en la Inspector para ver la definición de Lenguaje de estados de Amazon (ASL) de la máquina de estado, que se genera automáticamente en función de las selecciones que realice en las pestañas Acciones y Flujo y el panel Inspector.

> **Tip**
>
> Para editar la definición, puede abrir el editor de código seleccionando Código en la parte superior de la página. Para este tutorial, continúe con la definición generada automáticamente.

2. Especifique un nombre para la máquina de estado. Para ello, selecciona el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`CreditCardWorkflow`**.

3. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

   > **Note**
   >
   > (Opcional) Step Functions crea automáticamente un rol de ejecución para la máquina de estado con los privilegios mínimos necesarios para invocar la función de Lambda `RandomNumberforCredit` y publicarla en el tema de Amazon SNS.

   Si ya ha creado un rol de IAM con los permisos correctos para su máquina de estado y desea utilizarlo, en Permisos, seleccione Elegir un rol existente y, a continuación, seleccione un rol de la lista. O seleccione Escribir un ARN de rol y, a continuación, proporcione un ARN para ese rol de IAM.

4. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 2: Agregar las políticas de IAM restantes

Dado que Step Functions no genera automáticamente los permisos para invocar las funciones de Lambda utilizadas en el estado `Parallel`, es necesario añadir la política necesaria.

Para agregar la política restante

1. En la CreditCardWorkflowpágina, elija la función de IAM para su máquina de estado para acceder a la consola de IAM. Añadirá los permisos necesarios para el resto de las funciones de Lambda en esta página.
2. Elija Agregar permisos y luego Adjuntar políticas.
3. En el cuadro de búsqueda, escriba **`AWSLambdaRole`** y, a continuación, pulse Entrar.
4. Elija AWSLambdaRoley, a continuación, elija Adjuntar políticas. Esta política se agrega ahora al rol de ejecución de su máquina de estado. Esta política le permite invocar cualquier función de Lambda en su máquina de estado.

Paso 3: Ejecutar la máquina de estado

Las ejecuciones de máquinas de estado son instancias en las que se ejecuta un flujo de trabajo para realizar tareas.

Para ejecutar la máquina de estado

1. En la CreditCardWorkflowpágina, elija Iniciar ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.
2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

> **Note**
>
> No es necesario proporcionar ninguna entrada para ejecutar esta máquina de estado. Sin embargo, si es necesario, puede especificar una entrada de ejecución en el área Entrada del cuadro de diálogo Iniciar ejecución para otras máquinas de estado. Para ver un ejemplo de cómo proporcionar una entrada de ejecución a una máquina de estados, consulte el paso 4: Iniciar una nueva ejecución del tutorial Aprenda a usar AWS Step Functions Workflow Studio.

b. Seleccione Iniciar ejecución.

3. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Tutorial 7: Configurar entrada y salida

Una ejecución de Step Functions recibe un texto JSON como entrada y transfiere dicha entrada al primer estado en el flujo de trabajo. Los estados individuales de un flujo de trabajo reciben datos JSON como entrada y normalmente pasan datos JSON como salida al siguiente estado. De forma predeterminada, los datos pasan de un estado al siguiente del flujo de trabajo, a menos que haya configurado la entrada o la salida para uno o más estados del flujo de trabajo. Comprender cómo fluye esta información de un estado a otro y aprender a filtrar y manipular estos datos resulta esencial para diseñar e implementar de forma eficaz los flujos de trabajo en Step Functions.

Step Functions proporciona varios filtros para controlar el flujo de datos de entrada y salida entre estados. Los siguientes filtros están disponibles para su uso en los flujos de trabajo:

> **Note**
>
> Según el caso de uso, es posible que no necesite aplicar todos estos filtros en los flujos de trabajo.

`InputPath`

Selecciona QUÉ parte de toda la carga de entrada se utilizará como entrada de una tarea. Si especifica este campo, Step Functions lo aplicará en primer lugar.

`Parámetros`

Especifica CÓMO debe mostrarse la entrada antes de invocar la tarea. Con el campo `Parameters` puede crear una colección de pares clave-valor que se pasan como entrada a una integración de Servicio de AWS, por ejemplo, a una función AWS Lambda. Estos valores pueden ser estáticos o seleccionarse dinámicamente desde la entrada de estado o desde el objeto de contexto del flujo de trabajo.

`ResultSelector`

Determina QUÉ elegir del resultado de una tarea. Con el campo `ResultSelector` puede crear una colección de pares clave-valor que sustituyan el resultado de un estado y que pasen esa colección a `ResultPath`.

`ResultPath`

Determina DÓNDE colocar el resultado de una tarea. Utilice `ResultPath` para determinar si la salida de un estado es una copia de su entrada, el resultado que produce o una combinación de ambos.

`OutputPath`

Determina QUÉ enviar al siguiente estado. Con `OutputPath` puede filtrar la información no deseada y pasar solo la parte de datos JSON que le interese.

> **Tip**
>
> Los filtros `Parameters` y `ResultSelector` funcionan creando JSON, mientras que los filtros `InputPath` y `OutputPath` funcionan filtrando nodos específicos dentro de un objeto de datos JSON, y el filtro `ResultPath` funciona creando un campo en el que se puede añadir la salida.

En este tutorial aprenderá a realizar las siguientes tareas:

- Seleccione partes específicas de la entrada sin procesar mediante el InputPath filtro
- Manipular la entrada seleccionada mediante el filtro Parámetros
- Configurar la salida mediante los filtros ResultSelector, ResultPath y OutputPath

Para obtener más información acerca de la configuración de entrada y salida en los flujos de trabajo, consulte Procesamiento de entrada y salida en Step Functions.

Seleccione partes específicas de la entrada sin procesar mediante el InputPath filtro

Utilice el filtro `InputPath` para seleccionar una parte específica de la carga útil de entrada.

Si no especifica `InputPath`, su valor predeterminado es `$`, lo que hace que la tarea del estado se refiera a toda la entrada sin procesar en lugar de a una parte específica.

Para aprender a utilizar el filtro InputPath, realice los siguientes pasos:

- Paso 1: Crear una máquina de estado

- Paso 2: Ejecutar la máquina de estado
- Paso 3: Utilizar el filtro InputPath para seleccionar partes específicas de una entrada de ejecución

Paso 1: Crear una máquina de estado

> **Important**
>
> Asegúrese de que su máquina de estado esté en la misma AWS cuenta y región que la función Lambda que creó anteriormente.

1. Utilice el ejemplo de estado `Parallel` aprendido en el Tutorial 4 para crear una nueva máquina de estado. Asegúrese de que el prototipo de flujo de trabajo se parezca al siguiente prototipo.
2. Configure las integraciones para las funciones de lambda `check-identity` y `check-address`. Para obtener información sobre la creación de las funciones de Lambda y su uso en la máquina de estado, consulte Paso 1: Crear las funciones de Lambda para realizar las comprobaciones necesarias y Paso 2: Actualizar el flujo de trabajo: añadir tareas paralelas a realizar.
3. Para Carga, asegúrese de mantener la selección predeterminada de Usar entrada de estado como carga.
4. Seleccione Siguiente y, a continuación, realice los pasos del 1 a 3 en Paso 1: Guardar la máquina de estado del Tutorial 5 para crear una nueva máquina de estado. Para este tutorial, asigne a su máquina de estado el nombre **`WorkflowInputOutput`**.

Paso 2: Ejecutar la máquina de estado

1. En la WorkflowInputOutputpágina, elija Iniciar ejecución.
2. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar

un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

3. En el área de entrada, añada los siguientes datos JSON como entrada de ejecución.

```
{
  "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. Seleccione Iniciar ejecución.
5. La ejecución de la máquina de estado produce un error porque no ha especificado qué partes de la ejecución deben utilizar la entrada `check-identity` y las funciones de Lambda `check-address` para realizar la verificación de identidad y dirección requerida.
6. Continúe con el paso 3 de este tutorial para corregir el error.

Paso 3: Utilizar el filtro **`InputPath`** para seleccionar partes específicas de una entrada de ejecución

1. En la página Detalle de ejecución, seleccione Editar máquina de estado.
2. Para verificar la identidad del solicitante tal como se menciona en la entrada de ejecución proporcionada enPaso 2: Ejecutar la máquina de estado, edite la definición de la tarea de verificación de identidad de la siguiente manera:

```
...
{
    "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
      }
    }
  }
  ...
```

En consecuencia, los siguientes datos de JSON pasan a estar disponibles como entrada para la función `check-identity`.

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

3. Para verificar la dirección del solicitante mencionada en la entrada de ejecución, edite la definición de la tarea `Verify address` del siguiente modo:

```
...
{
    "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
      }
```

```
    }
  }
  ...
```

En consecuencia, los siguientes datos de JSON pasan a estar disponibles como entrada para la función `check-address`.

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

4. Seleccione Iniciar ejecución. La ejecución de la máquina de estado ahora se completa correctamente.

Manipular la entrada seleccionada mediante el filtro Parámetros

Si bien el filtro `InputPath` le ayuda a limitar la entrada JSON sin procesar que proporciona, con el filtro `Parameters` puede pasar una colección de pares clave-valor como entrada. Estos pares clave-valor pueden ser valores estáticos definidos en la definición de la máquina de estado o valores seleccionados de la entrada bruta mediante `InputPath`.

En sus flujos de trabajo, se aplican `Parameters` después de `InputPath`. `Parameters` ayuda a especificar cómo la tarea subyacente acepta su carga útil de entrada. Por ejemplo, si la función de Lambda `check-address` acepta un parámetro de cadena como entrada en lugar de los datos JSON, puede usar el filtro `Parameters` para transformar la entrada.

En el siguiente ejemplo, el filtro `Parameters` recibe la entrada que ha seleccionado `InputPath` en Paso 3: Utilizar el filtro `InputPath` para seleccionar partes específicas de una entrada de ejecución y aplica la función intrínseca `States.Format` a los elementos de entrada para crear una cadena llamada `addressString`. Las funciones intrínsecas le ayudan a realizar operaciones básicas de procesamiento de datos en una entrada determinada. Para obtener más información, consulte Funciones intrínsecas.

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

```
  }
```

En consecuencia, se crea la siguiente cadena y se proporciona a la función de Lambda `check-address` como entrada.

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

Configurar la salida mediante los filtros ResultSelector, ResultPath y OutputPath

Cuando se invoca la función de Lambda de `check-address` en la máquina de estado WorkflowInputOutput, la función devuelve una carga útil de salida después de realizar la verificación de dirección. En la página de Detalles de la ejecución, elija el paso Verificar dirección y vea la carga útil de salida dentro de Resultado de la tarea en el panel Detalles del paso.

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

Utilizar ResultSelector

Si necesita proporcionar ahora el resultado de las comprobaciones de identidad y dirección en los siguientes estados del flujo de trabajo, puede seleccionar el nodo Payload.body en el JSON de salida y utilizar el filtro de función intrínseca `StringToJson` en el filtro `ResultSelector` para formatear los datos si es necesario.

`ResultSelector` selecciona lo que se necesita del resultado de la tarea. En el siguiente ejemplo, `ResultSelector` toma la cadena de $.Payload.body y aplica la función intrínseca `States.StringToJson` para convertir la cadena a JSON y coloca el JSON resultante dentro del nodo de identidad.

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

De este modo se crean los siguientes datos JSON.

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

Mientras trabaja con estos filtros de entrada y salida, es posible que también se produzcan errores de tiempo de ejecución debido a la especificación de expresiones de ruta JSON no válidas. Para obtener más información, consulte.

Utilizar ResultPath

Puede especificar una ubicación en la carga útil de entrada inicial para guardar el resultado del procesamiento de tareas de un estado mediante el campo `ResultPath`. Si no especifica `ResultPath`, su valor predeterminado es $, lo que hace que la carga útil de entrada inicial se sustituya por el resultado sin procesar de la tarea. Si especifica `ResultPath` como `null`, el resultado sin procesar se descarta y la carga útil de entrada inicial pasa a ser la salida efectiva.

Si aplica el campo `ResultPath` a los datos JSON creados con el campo `ResultSelector`, el resultado de la tarea se añade dentro del nodo de resultados de la carga útil de entrada, como se muestra en el siguiente ejemplo:

```
{
  "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
    }
  }
}
```

Utilizar OutputPath

Puede seleccionar una parte de la salida de estado después de la aplicación de `ResultPath` para pasar al siguiente estado. Esto le permite filtrar la información no deseada y pasar solo la parte de JSON que le interese.

En el siguiente ejemplo, el campo `OutputPath` guarda la salida de estado dentro del nodo de resultados: `"OutputPath": "$.results"`. De este modo, el resultado final del estado, que puede pasar al siguiente estado, es el siguiente:

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

Utilizar las características de la consola para visualizar los flujos de datos de entrada y salida

Puede visualizar el flujo de datos de entrada y salida entre los estados de los flujos de trabajo mediante el simulador de flujos de datos de la consola de Step Functions o la opción de vista avanzada de la página Detalles de ejecución.

Tutorial 8: Depurar errores en la consola

Al trabajar con Step Functions, es posible que se produzcan errores de tiempo de ejecución por motivos como los siguientes:

- Una ruta JSON no válida para el campo Variable en el estado `Choice`.
- Problema de definición de la máquina de estado, por ejemplo, no se ha definido ninguna regla de correspondencia para un estado `Choice`.
- Expresiones de ruta JSON no válidas al aplicar filtros para manipular la entrada y la salida.
- Fallos en las tareas debido a una excepción de la función de Lambda.
- Errores de permiso de IAM.

En este tutorial, aprenderá a depurar algunos de estos errores mediante la consola de Step Functions. Para obtener más información, consulte Control de errores en Step Functions.

Temas

Depuración de la ruta no válida: error de elección de estado

Si especifica una ruta JSON incorrecta o irresoluble en el campo Variable del estado `Choice` o no define una regla coincidente en el estado `Choice`, recibe un error al ejecutar el flujo de trabajo.

Para ilustrar el error de ruta no válida, en este tutorial se presenta un error de estado `Choice` en el flujo de trabajo. Utilizará la máquina de estado CreditCardWorkflow y editará su definición para introducir el error.

1. Abra la consola de Step Functions y elija Crear la máquina de estado CreditCardWorkflow.
2. Seleccione Editar para editar la definición de la máquina de estado. Realice el cambio resaltado en el siguiente código en la definición de su máquina de estado.

```
{
  "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. Seleccione Guardar y, a continuación, seleccione Guardar de todos modos.
4. Ejecutar la máquina de estado.
5. En la página de detalles de ejecución de la máquina de estado, lleve a cabo alguna de las siguientes operaciones:

   a. Seleccione Causa en el mensaje de error para ver el motivo del error de ejecución.

   b. Seleccione Mostrar detalles de pasos en el mensaje de error para ver el paso que provocó el error.
6. En la pestaña Entrada y salida de la sección Detalles de pasos, pulse el botón de alternancia de Vista avanzada para ver la ruta de transferencia de datos de entrada y salida de un estado seleccionado.
7. En Vista gráfica, asegúrese de que se seleccione ¿Crédito aplicado >= 5000? y proceda del modo siguiente:

   a. Vea el valor de entrada del estado en el cuadro Entrada.

b. Seleccione la pestaña Definición y observe la ruta JSON especificada para el campo Variable.

   El valor de entrada del estado Credit applied **>=** 5000? es un valor numérico, mientras que ha especificado la ruta JSON para el valor de entrada como `$.Payload`. Durante la ejecución de la máquina de estado, el estado `Choice` no puede resolver esta ruta JSON porque no existe.

8. Edite la máquina de estado para especificar el valor del campo Variable como**$**.

```
{
  "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"
    },
    ...
    ...
  }
}
```

Depurar los errores de expresión de la ruta JSON al aplicar filtros de entrada y salida

Mientras trabaja con los filtros de entrada y salida, es posible que se produzcan errores de tiempo de ejecución debido a la especificación de expresiones de ruta JSON no válidas.

En el siguiente ejemplo, se utiliza la máquina de estado WorkflowInputOutput creado en el Tutorial 5 y se muestra un escenario en el que se utiliza el filtro `ResultSelector` para seleccionar partes del resultado de la tarea.

1. Aplique el filtro `ResultSelector` para elegir una parte del resultado de la tarea para el paso Verificar identidad. Para ello, edite la definición de la máquina de estado de la siguiente manera:

```
{
  "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. Ejecutar la máquina de estado.
3. En la página de detalles de ejecución de la máquina de estado, lleve a cabo las siguientes operaciones:

a. Seleccione Causa en el mensaje de error para ver el motivo del error de ejecución.

b. Seleccione Mostrar detalles de pasos en el mensaje de error para ver el paso que provocó el error.

4. En el mensaje de error, tenga en cuenta que el contenido del nodo $.Payload.body es una cadena JSON de escape. El error se ha producido porque no se puede hacer referencia a una cadena mediante la notación de ruta JSON.

5. Para hacer referencia al nodo $.payload.BODY.MESSAGE, haga lo siguiente:

   a. Utilice la función intrínseca `States.StringToJSON` para convertir primero la cadena a un formato JSON.

   b. Especifique la ruta JSON para el nodo $.Payload.body.message dentro de la función intrínseca.

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

6. Ejecute de nuevo la máquina de estado.

Casos de uso

Con AWS Step Functions podrá crear flujos de trabajo visuales que ayudan a convertir rápidamente los requisitos empresariales en aplicaciones. Step Functions gestiona el estado, los puntos de control y los reinicios, y proporciona capacidades integradas para tratar automáticamente los errores y las excepciones. Para comprender mejor las capacidades que Step Functions puede proporcionarle, lea los siguientes casos de uso:

Temas

Procesamiento de datos

A medida que crece el volumen de datos procedentes de orígenes cada vez más diversos, las organizaciones se dan cuenta de que necesitan procesar estos datos con rapidez y garantizar que toman decisiones empresariales más rápidas y con fundamento. Para procesar los datos a escala, las organizaciones deben aprovisionar recursos de manera elástica para gestionar la información que reciben de los dispositivos móviles, las aplicaciones, los satélites, el marketing y las ventas, los almacenes de datos operativos y la infraestructura, entre otros.

Step Functions proporciona la escalabilidad, la fiabilidad y la disponibilidad necesarias para gestionar correctamente los flujos de trabajo de procesamiento de datos. Con Step Functions, puede gestionar millones de ejecuciones simultáneas, ya que se escala horizontalmente y proporciona flujos de trabajo tolerantes a errores. Procese los datos más rápido mediante ejecuciones paralelas, como el tipo de estado `Parallel` de Step Functions, o el paralelismo dinámico, mediante su tipo de estado `Map`. Como parte de su flujo de trabajo, puede usar el estado `Map` para recorrer objetos en iteración en un almacén de datos estáticos, como un bucket de Amazon S3. Step Functions también le facilita volver a intentar ejecuciones que producen error o elegir una forma específica de tratar los errores sin necesidad de gestionar un proceso complejo.

Dependiendo de sus necesidades de procesamiento de datos, Step Functions se integra directamente con otros servicios de procesamiento de datos proporcionados por AWS, por ejemplo,

AWS Batch para el procesamiento por lotes, Amazon EMR para el procesamiento de macrodatos, AWS Glue para la preparación de datos, Athena para el análisis de datos y AWS Lambda para la computación.

Entre los ejemplos de los tipos de flujos de trabajo de procesamiento de datos para los que los clientes utilizan Step Functions se incluyen:

Procesamiento de archivos, vídeos e imágenes

- Tome una conjunto de archivos de vídeo y conviértalos a otros tamaños o resoluciones que sean ideales para el dispositivo en el que se van a mostrar, como teléfonos móviles, ordenadores portátiles o televisiones.
- Tome una gran colección de fotos subidas por los usuarios y conviértalas en miniaturas o imágenes de varias resoluciones que luego puedan mostrarse en los sitios web de los usuarios.
- Tome datos semiestructurados, como un archivo CSV, y combínelos con datos no estructurados, como una factura, para elaborar un informe comercial que se envíe mensualmente a las partes interesadas de la empresa.
- Tome los datos de observación de la Tierra recopilados por los satélites, conviértalos en formatos que se alineen entre sí y, a continuación, agregue otros orígenes de datos recopilados en la Tierra para obtener información adicional.
- Tome los registros de transporte de los distintos modos de transporte para los productos y busque optimizaciones mediante simulaciones Monte Carlo. Posteriormente, envíe los informes a las organizaciones y personas que confían en usted para enviar sus mercancías.

Coordine trabajos de extracción, transformación y carga (ETL):

- Combine registros de oportunidades de venta con conjuntos de datos de métricas de marketing mediante una serie de pasos de preparación de datos usando AWS Glue y elabore informes de inteligencia empresarial que se puedan utilizar en toda la organización.
- Cree, inicie y finalice un clúster de Amazon EMR para el procesamiento de macrodatos.

Procesamiento por lotes y cargas de trabajo de computación de alto rendimiento (HPC):

- Cree canalizaciones de análisis secundarios de genómica que procesen secuencias genómicas completas sin procesar y las conviertan en llamadas variantes. Alinee archivos sin procesar con una secuencia de referencia y llame a variantes en una lista especificada de cromosomas mediante el paralelismo dinámico.

- Encuentre eficiencias en la producción de su próximo dispositivo móvil u otros equipos electrónicos mediante la simulación de varios diseños con diferentes compuestos eléctricos y químicos. Proceso sus cargas de trabajo en grandes lotes mediante diversas simulaciones para obtener el diseño óptimo.

Machine learning

El machine learning permite a las organizaciones analizar con rapidez los datos recopilados para identificar patrones y, posteriormente, tomar decisiones con la mínima intervención humana. El machine learning comienza con un conjunto inicial de datos, conocido como datos de entrenamiento. Estos datos de entrenamiento ayudan a aumentar la precisión de las predicciones de un modelo de machine learning y sirven de base para el aprendizaje de este modelo. El modelo se implementa en producción una vez que se considera lo bastante preciso como para satisfacer las necesidades de negocio. El kit de desarrollo de software (SDK) para ciencia de datos de AWS Step Functions es una biblioteca de código abierto con la que puede crear fácilmente flujos de trabajo que preprocesan datos, entrenan y luego publican sus modelos con Amazon SageMaker y Step Functions.

El preprocesamiento de los conjuntos de datos existentes es la forma en que una organización suele crear los datos de entrenamiento. Este método agrega información, por ejemplo, al etiquetar objetos en una imagen, al anotar texto o al procesar audio. Para preprocesar los datos, puede utilizar AWS Glue o bien crear una instancia con bloc de notas de SageMaker que ejecute la aplicación de cuaderno de Jupyter. Una vez que los datos estén listos, podrá cargarlos en Amazon S3 para acceder a ellos con facilidad. A medida que se entrenan los modelos de machine learning, puede realizar ajustes en los parámetros de cada modelo para mejorar la precisión hasta que esté listo para implementarse.

Step Functions le permite orquestar flujos de trabajo integrales de machine learning en SageMaker. Estos flujos de trabajo pueden incluir el preprocesamiento y el posprocesamiento de datos, la ingeniería de características, la validación de datos y la evaluación de modelos. Una vez que el modelo se haya implementado en producción, podrá perfeccionar y probar nuevos enfoques para mejorar continuamente los resultados empresariales. Puede crear flujos de trabajo listos para la producción directamente en Python, o bien puede usar el SDK de Step Functions Data Science para copiar ese flujo de trabajo, experimentar con nuevas opciones y poner el flujo de trabajo perfeccionado en producción.

Algunos tipos de flujos de trabajo de machine learning para los que los clientes utilizan Step Functions incluyen:

Detección del fraude

- Identifique y evite que se produzcan transacciones fraudulentas, como el fraude crediticio.
- Detecte y evite la apropiación de cuentas mediante modelos de machine learning entrenados.
- Identifique el abuso promocional, incluida la creación de cuentas falsas, para que pueda tomar medidas con rapidez.

Personalización y recomendaciones

- Recomiende productos a clientes objetivo en función de lo que se prevé que atraiga su interés.
- Prediga si un cliente actualizará su cuenta de un nivel gratuito a una suscripción de pago.

Enriquecimiento de datos

- Utilice el enriquecimiento de datos como parte del preprocesamiento para proporcionar mejores datos de entrenamiento para modelos de machine learning más precisos.
- Anote fragmentos de texto y audio para agregar información sintáctica, como sarcasmo y jerga.
- Etiquete objetos adicionales en las imágenes para proporcionar información crítica de la que pueda aprender el modelo, por ejemplo, si se trata de una manzana, un balón de baloncesto, una roca o un animal.

Orquestación de microservicios

La arquitectura de microservicios divide las aplicaciones en servicios con acoplamiento flexible. Entre las ventajas se incluyen una escalabilidad mejorada, una mayor resiliencia y un tiempo de comercialización más rápido. Cada microservicio es independiente, de modo que resulta sencillo escalar verticalmente un único servicio o función sin necesidad de escalar toda la aplicación. Los servicios individuales se acoplan de forma flexible, lo que permite a los equipos independientes centrarse en un único proceso de negocio, sin necesidad de comprender toda la aplicación.
Los microservicios también le permiten elegir qué componentes individuales se adaptan a sus necesidades de negocio, lo que le aporta la flexibilidad de cambiar su selección sin tener que reescribir todo su flujo de trabajo. Los distintos equipos pueden usar los marcos y lenguajes de programación que prefieran para trabajar con su microservicio, y este microservicio puede seguir comunicándose con cualquier otro de la aplicación a través de las interfaces de programación de aplicaciones (API).

Step Functions le ofrece varias formas de gestionar sus flujos de trabajo de microservicios. Para flujos de trabajo de larga duración, puede utilizar los flujos de trabajo estándar con la integración de AWS Fargate para orquestar las aplicaciones que se ejecutan en contenedores. En el caso de flujos de trabajo de corta duración y gran volumen que requieren una respuesta inmediata, los flujos de trabajo rápidos sincrónicos son ideales. Se pueden usar para aplicaciones móviles o basadas en la web, que suelen tener flujos de trabajo de corta duración y requieren que se completen una serie de pasos antes de obtener una respuesta. Puede activar directamente un flujo de trabajo rápido sincrónico desde Amazon API Gateway y la conexión se mantendrá abierta hasta que el flujo de trabajo se complete o se agote el tiempo de espera. Para flujos de trabajo de corta duración que no requieren una respuesta inmediata, Step Functions ofrece flujos de trabajo rápidos asincrónicos.

Entre los ejemplos de algunas orquestaciones de API que utilizan Step Functions se incluyen:

Flujos de trabajo sincrónicos o en tiempo real

- Cambie un valor de un registro, como actualizar el apellido de un empleado, y haga que el cambio se vea de inmediato en la pantalla.
- Actualice un pedido durante el proceso de pago, por ejemplo, agregando, quitando o cambiando la cantidad de un artículo, y envíe inmediatamente la actualización al cliente.
- Ejecute un trabajo de procesamiento rápido y devuelva inmediatamente el resultado al solicitante.

Orquestación de contenedores

- Ejecute trabajos en Kubernetes con Amazon Elastic Kubernetes Service o en Amazon Elastic Container Service (ECS) con Fargate e intégrelos con otros servicios de AWS, como el envío de notificaciones con Amazon SNS, como parte del mismo flujo de trabajo.

Automatización de TI y seguridad

La automatización de la TI puede ayudar a gestionar operaciones cada vez más complejas y lentas, como actualizar y aplicar parches al software, implementar actualizaciones de seguridad para abordar las vulnerabilidades, seleccionar la infraestructura, sincronizar los datos, enrutar los tickets de soporte, etc. La automatización de las tareas repetitivas y lentas puede permitir a su organización completar las operaciones rutinarias de forma rápida y coherente a gran escala. De este modo, puede centrarse en el trabajo estratégico, como el desarrollo de características, las solicitudes de soporte complejas y la innovación, al tiempo que satisface estas crecientes demandas.

Step Functions le permite crear flujos de trabajo que se escalan automáticamente para satisfacer las necesidades de su empresa sin necesidad de intervención manual. En los casos en que se produce un error en el flujo de trabajo, no suele ser necesaria una intervención manual. Step Functions te permite reintentar las tareas con errores automáticamente y un retroceso exponencial que puede gestionar los errores en tu flujo de trabajo.

Puede haber situaciones en las que sea necesaria la intervención humana antes de que el flujo de trabajo pueda avanzar. Por ejemplo, la aprobación de un aumento sustancial del crédito puede requerir la aprobación humana. Para gestionarlo, puede definir una lógica de ramificación en Step Functions, de modo que solo las solicitudes que superen un importe definido requieran la aprobación humana, mientras que el resto de solicitudes se completan automáticamente. En los casos en los que se requiere la aprobación humana, Step Functions le permite pausar el flujo de trabajo en un paso específico, esperar una respuesta y continuar con el flujo de trabajo una vez recibida la respuesta.

Entre los ejemplos de los tipos de flujos de trabajo de automatización para los que los clientes utilizan Step Functions se incluyen:

Automatización de TI

- Solucione automáticamente incidentes como la apertura de un puerto SSH, la falta de espacio en disco o el acceso público a un bucket de Amazon S3.
- Automatice la implementación de AWS CloudFormation StackSets

Automatización de seguridad

- Automatice la respuesta a un escenario en el que un usuario y su clave de acceso estén expuestos.
- Solucione automáticamente las respuestas a incidentes de seguridad de acuerdo con las acciones de política definidas, como restringir las acciones a ARN específicas o aplicar otras acciones.
- Advierta a los empleados de los correos electrónicos de phishing (suplantación de identidad) unos segundos después de recibirlos.

Aprobación humana

- Automatice el entrenamiento de un modelo de machine learning y, posteriormente, solicite la aprobación manual del modelo por parte de un científico de datos antes de implementarlo o rechazarlo automáticamente según la respuesta recibida.

- Automatice el enrutamiento de los comentarios de los clientes recibidos en función del análisis de sus opiniones, de modo que quienes tengan una opinión negativa se escalen de inmediato para revisarlos manualmente.

Cómo funciona Step Functions

En esta sección, se describen conceptos importantes que le ayudarán a familiarizarse con AWS Step Functions y a entender cómo funciona.

Temas

Flujos de trabajo estándar en comparación con flujos de trabajo rápidos

Al crear una nueva máquina de estado, debe seleccionar un Tipo que sea Estándar o bien Rápido. El Tipo predeterminado para las máquinas de estado es Estándar. Una máquina de estado cuyo Tipo es Estándar se denomina Flujo de trabajo estándar y una máquina de estado cuyo Tipo es Rápido se denomina Flujo de trabajo rápido.

Se define la máquina de estados mediante Lenguaje de estados de Amazon tanto para los flujos de trabajo estándar como rápidos. Las ejecuciones de la máquina de estado se comportarán de forma diferente en función del Type que seleccione.

> **Important**
>
> El Tipo que seleccione no se podrá cambiar después de que haya creado la máquina de estado.

> **Note**
>
> Si define las máquinas de estado fuera de la consola de Step Functions, por ejemplo, en un editor de su elección, debe guardar las definiciones de las máquinas de estado con la extensión .asl.json.

Los flujos de trabajo estándar son ideales para flujos de trabajo auditables, duraderos y de ejecución prolongada (hasta un año). Puede recuperar el historial de ejecuciones completo mediante la API de Step Functions, hasta 90 días después de que la ejecución se complete. Los flujos de trabajo estándar siguen un modelo de exactamente una vez en el que las tareas y los estados nunca se ejecutan más de una vez a no ser que haya especificado el comportamiento `Retry` en ASL. Esto hace que los flujos de trabajo estándar sean adecuados para orquestar acciones que no sean idempotentes, como el inicio de un clúster de Amazon EMR o el procesamiento de pagos. Las ejecuciones de flujos de trabajo estándar se facturan de acuerdo con el número de transiciones de estado que se han procesado.

Los flujos de trabajo rápidos son ideales para cargas de trabajo de procesamiento de eventos de un volumen elevado como la incorporación de datos de IoT, el streaming de la transformación y el procesamiento de los datos y los backends de aplicaciones móviles. Pueden ejecutarse durante un máximo de 5 minutos. Los flujos de trabajo rápidos emplean un modelo de al menos una vez en el que existe la posibilidad de que una ejecución se ejecute más de una vez. Esto hace que los flujos de trabajo rápido sean ideales para orquestar acciones idempotentes como la transformación de los datos de entrada y el almacenamiento a través de PUT en Amazon DynamoDB. Las ejecuciones de flujos de trabajo rápidos se facturan según el número de ejecuciones, la duración de la ejecución y la memoria consumida mientras tuvo lugar la ejecución.

Los flujos de trabajo estándar y rápidos se pueden iniciar automáticamente como respuesta a eventos como solicitudes HTTP a través de Amazon API Gateway (API completamente administradas a escala), reglas de IoT y más de 140 orígenes de eventos en Amazon EventBridge.

> **Tip**
>
> Para implementar un ejemplo de flujo de trabajo rápido en su Cuenta de AWS, consulte el Módulo 7: API Gateway, estado Parallel, flujos de trabajo rápidos de El workshop de AWS Step Functions.

Para obtener información sobre la experiencia de consola para las ejecuciones de flujos de trabajo estándar y rápidos, consulte Ejecuciones de flujos de trabajo estándar y rápidos en la consola.

Flujos de trabajo estándar en comparación con flujos de trabajo rápidos

	Flujos de trabajo estándar	Flujos de trabajo rápidos: síncronos y asíncronos
Duración máxima	Un año	Cinco minutos
Velocidad de inicio de ejecución admitida	Para obtener información sobre las cuotas relacionadas con la velocidad de inicio de ejecución admitida, consulte Cuotas relacionadas con la limitación controlada de las acciones de la API.	Para obtener información sobre las cuotas relacionadas con la velocidad de inicio de ejecución admitida, consulte Cuotas relacionadas con la limitación controlada de las acciones de la API.
Velocidad de transición de estado admitida	Para obtener información sobre las cuotas relacionadas con la velocidad de transición de estado admitida, consulte Cuotas relacionadas con la limitación controlada de estados.	Sin límite
Precios	El precio se basa en el número de transiciones de estado. Una transición de estado se cuenta cada vez	El precio varía en función del número de ejecuciones que realice, la duración y el consumo de memoria.

	Flujos de trabajo estándar	Flujos de trabajo rápidos: síncronos y asíncronos
	que se completa un paso de la ejecución.	
Historial de ejecuciones	Las ejecuciones se pueden enumerar y describir con API de Step Functions. Las ejecuciones se pueden depurar visualmente a través de la consola. También se pueden analizar en CloudWatch Logs mediante la habilitación del registro en la máquina de estado. Para obtener más información sobre cómo depurar las ejecuciones de flujos de trabajo estándar en la consola, consulte Ejecuciones de flujos de trabajo estándar y rápidos en la consola y Visualización y depuración de ejecución.	El historial de ejecuciones es ilimitado, es decir, se conservan tantas entradas del historial de ejecuciones como se puedan generar en un período de 5 minutos. Las ejecuciones se pueden inspeccionar en CloudWatch Logs mediante la habilitación del registro en la consola de Step Functions. Para obtener más información sobre cómo depurar las ejecuciones de flujos de trabajo rápidos en la consola, consulte Ejecuciones de flujos de trabajo estándar y rápidos en la consola y Visualización y depuración de ejecución.
Semántica de ejecuciones	Ejecución del flujo de trabajo exactamente una vez.	Flujos de trabajo rápidos asíncronos: ejecución de flujo de trabajo al menos una vez. Flujos de trabajo rápidos síncronos: ejecución de flujo de trabajo como máximo una vez.

	Flujos de trabajo estándar	Flujos de trabajo rápidos: síncronos y asíncronos
Integraciones de servicios	Admite todos los patrones y las integraciones de servicios.	Admite todas las integraciones de servicios. **Note** Los flujos de trabajo rápidos no admiten patrones de integración de servicios Job-run (`.sync`) o Callback (`.waitForTaskToken`).
Step Functions	Admite actividades de Step Functions.	No es compatible con actividades de Step Functions.

Flujos de trabajo rápidos síncronos y asíncronos

Puede elegir entre dos tipos de flujos de trabajo rápidos: flujos de trabajo rápidos asíncronos y flujos de trabajo rápidos síncronos.

- Los flujos de trabajo rápidos asíncronos devuelven una confirmación de que el flujo de trabajo se ha iniciado, pero no esperan a que se complete. Para obtener el resultado, debe sondear los CloudWatch Logs del servicio. Puede utilizar flujos de trabajo rápidos asíncronos cuando no necesite una salida de respuesta inmediata, como en los servicios de mensajería o en el procesamiento de datos del que no dependen otros servicios. Puede iniciar flujos de trabajo rápidos asíncronos en respuesta a un evento, mediante un flujo de trabajo anidado en Step Functions o mediante la llamada a la API `StartExecution`.
- Los flujos de trabajo rápidos síncronos inician un flujo de trabajo, esperan a que se complete y, a continuación, devuelven el resultado. Los flujos de trabajo rápidos síncronos se pueden utilizar para orquestar microservicios. Con flujos de trabajo rápidos síncronos, puede desarrollar aplicaciones sin necesidad de desarrollar código adicional para gestionar errores o reintentos o

ejecutar tareas paralelas. Puede ejecutar flujos de trabajo rápidos síncronos invocados desde Amazon API Gateway, AWS Lambda o mediante la llamada a la API `StartSyncExecution`.

> **Note**
>
> Si ejecuta flujos de trabajo rápidos de Step Functions de forma sincrónica desde la consola, la solicitud `StartSyncExecution` caduca después de 60 segundos. Para ejecutar flujos de trabajo rápidos de forma sincrónica durante un máximo de cinco minutos, realice la solicitud `StartSyncExecution` mediante el SDK de AWS o AWS Command Line Interface (AWS CLI) en lugar de la consola de Step Functions.

Las llamadas a la API de ejecución rápidas sincrónica no contribuyen a los límites de capacidad de las cuentas existentes. Step Functions proporciona capacidad bajo demanda y escala automáticamente con una carga de trabajo sostenida. Los picos de carga de trabajo pueden reducirse hasta que haya capacidad disponible.

Garantías de ejecución

Flujos de trabajo estándar	Flujos de trabajo rápidos asíncronos	Flujos de trabajo rápidos síncronos	
Ejecución del flujo de trabajo exactamente una vez	Ejecución del flujo de trabajo al menos una vez	Ejecución del flujo de trabajo como máximo una vez	
El estado de ejecución persiste internamente entre las transiciones de estado.	El estado de ejecución no persiste entre transiciones de estado.	El estado de ejecución no persiste entre transiciones de estado.	
Devuelve automáticamente una respuesta idempotente al iniciar una ejecución con el mismo nombre que	La idempotencia no se gestiona automáticamente. Si se inician varios flujos de trabajo con el mismo	La idempotencia no se gestiona automáticamente. Step Functions espera una vez que se inicia una	

Flujos de trabajo estándar	Flujos de trabajo rápidos asíncronos	Flujos de trabajo rápidos síncronos	
un flujo de trabajo en ejecución. El nuevo flujo de trabajo no se inicia y se produce una excepción una vez que se completa el flujo de trabajo que se está ejecutando actualmente.	nombre, se producen ejecuciones simultáneas. Puede provocar la pérdida del estado del flujo de trabajo interno si la lógica de la máquina de estados no es idempotente.	ejecución y devuelve el resultado de la máquina de estado al finalizar. Los flujos de trabajo no se reinician si se produce una excepción.	

Flujos de trabajo estándar	Flujos de trabajo rápidos asíncronos	Flujos de trabajo rápidos síncronos	
Los datos del historial de ejecuciones se eliminan después de 90 días. Los nombres de los flujos de trabajo se pueden reutilizar después de eliminar los datos de ejecución desactualizados. Para cumplir con los requisitos de conformidad, organizativos o normativos, se puede reducir el periodo de retención del historial de ejecución a 30 días mediante el envío de una solicitud de cuota. Para ello, utilice AWS Support Center Console y cree un caso nuevo.	Step Functions no captura el historial de ejecución. El registro se debe habilitar a través de Registros de Amazon CloudWatch.	Step Functions no captura el historial de ejecución. El registro se debe habilitar a través de Registros de Amazon CloudWatch.	

Optimización de costes mediante flujos de trabajo rápidos

Step Functions determina los precios de los flujos de trabajo estándar y rápidos según el tipo de flujo de trabajo que utilice para crear sus máquinas de estados. Para optimizar el coste de sus flujos de trabajo sin servidor, puede seguir una de las siguientes recomendaciones o ambas:

Temas

- Consejo n.º 1: Anidar flujos de trabajo rápidos dentro de flujos de trabajo estándar
- Consejo n.º 2: Convertir flujos de trabajo estándar en flujos de trabajo rápidos

Para obtener información sobre cómo afecta a la facturación la elección de un tipo de flujo de trabajo estándar o rápidos, consulte Precios de AWS Step Functions.

Consejo n.º 1: Anidar flujos de trabajo rápidos dentro de flujos de trabajo estándar

Step Functions ejecuta flujos de trabajo que tienen una duración y un número de pasos finitos. Algunos flujos de trabajo pueden completar la ejecución en un breve periodo de tiempo. Otros pueden requerir una combinación de flujos de trabajo de larga duración y de alta frecuencia de eventos. Con Step Functions, puede crear flujos de trabajo grandes y complejos a partir de varios flujos de trabajo más pequeños y sencillos.

Por ejemplo, para crear un flujo de trabajo de procesamiento de pedidos, puede incluir todas las acciones no idempotentes en un flujo de trabajo estándar. Esto podría incluir acciones como la aprobación de pedidos mediante la interacción humana y el procesamiento de pagos. A continuación, puede combinar una serie de acciones idempotentes, como enviar notificaciones de pago y actualizar el inventario de productos, en un flujo de trabajo rápido. Puede agrupar este flujo de trabajo rápido en el flujo de trabajo estándar. En este ejemplo, el flujo de trabajo estándar se conoce como máquina de estado principal. El flujo de trabajo rápido anidado se conoce como máquina de estado secundaria.

Consejo n.º 2: Convertir flujos de trabajo estándar en flujos de trabajo rápidos

Puede convertir sus flujos de trabajo estándar existentes en flujos de trabajo rápidos si cumplen los siguientes requisitos:

- El flujo de trabajo debe completar su ejecución en cinco minutos.
- El flujo de trabajo se ajusta a un modelo de ejecución de al menos una vez. Esto significa que cada paso del flujo de trabajo puede ejecutarse más de una vez exactamente.
- El flujo de trabajo no utiliza los patrones de integración de servicios `.waitForTaskToken` o `.sync`.

> **Important**
>
> Los flujos de trabajo rápidos utilizan Registros de Amazon CloudWatch para registrar los historiales de ejecución. Al utilizar CloudWatch Logs, se incurrirá en costes adicionales.

Convertir un flujo de trabajo estándar en un flujo de trabajo rápido mediante la consola

1. Abra la consola de Step Functions.
2. En la página Máquinas de estados, seleccione una máquina de estado de tipo estándar para abrirla.

   > **Tip**
   >
   > En la lista desplegable Cualquier tipo, seleccione Estándar para filtrar la lista de máquinas de estados y ver solo los flujos de trabajo estándar.

3. Seleccione Copiar en uno nuevo.

   Workflow Studio se abre en Modo Diseño mostrando el flujo de trabajo de la máquina de estado que ha seleccionado.
4. (Opcional) Actualice el diseño del flujo de trabajo.
5. Especifique un nombre para la máquina de estado. Para ello, elija el icono de edición situado junto al nombre predeterminado de máquina de estado de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.
6. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Asegúrese de elegir Rápido en Tipo. Mantenga el resto de selecciones predeterminadas en la Configuración de máquina de estado.

   > **Note**
   >
   > Si va a convertir un flujo de trabajo estándar previamente definido en AWS CDK o AWS SAM, debe cambiar el valor `Type` y el nombre de `Resource`.

7. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Para obtener más información sobre las prácticas recomendadas y las directrices al gestionar la optimización de costes de sus flujos de trabajo, consulte Creación de flujos de trabajo de AWS Step Functions rentables.

Estados

Los estados individuales pueden tomar decisiones en función de su entrada, realizar acciones a partir de esas entradas y transferir la salida a otros estados. En AWS Step Functions se definen los flujos de trabajo en Amazon States Language (ASL). La consola de Step Functions proporciona una representación gráfica de esa máquina de estado para ayudar a visualizar la lógica de la aplicación.

> **Note**
>
> Si define las máquinas de estado fuera de la consola de Step Functions, por ejemplo, en un editor de su elección, debe guardar las definiciones de las máquinas de estado con la extensión .asl.json.

Los estados son elementos de la máquina de estado. La referencia de los estados se realiza por su nombre, que puede ser cualquier cadena, pero que debe ser único dentro del ámbito de toda la máquina de estado.

Los estados pueden realizar una gran variedad de funciones en la máquina de estado:

- Puede realizar alguna tarea en la máquina de estado (un estado Task).
- Pueden tomar una decisión entre las ramificaciones de una ejecución (estado Choice).

- Pueden detener una ejecución con errores o con éxito (estado Fail o Succeed).
- Pueden transferir su entrada a su salida o insertar ciertos datos fijos en el flujo de trabajo (un estado Pass)
- Pueden proporcionar un retraso que dure cierto tiempo o hasta una fecha u hora determinada (estado Wait)
- Pueden iniciar ramificaciones de ejecución paralelas (estado Parallel).
- Pueden iterar pasos de forma dinámica (un estado Map)

A continuación, se muestra un ejemplo de un estado denominado `HelloWorld` que realiza una función de AWS Lambda.

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

Los estados comparten muchas características comunes:

- Un campo `Type` que indique de qué tipo de estado se trata.
- Un campo `Comment` opcional que contenga comentarios o descripciones del estado en lenguaje natural.
- Todos los estados (excepto `Succeed` o `Fail`) necesitan un campo `Next` o pueden adoptar un estado terminal a través de un campo `End`.

> **Note**
>
> Los estados `Choice` pueden tener uno o varios campos `Next`, pero solo puede haber uno en cada regla de Choice. Los estados `Choice` no pueden utilizar `End`.

Algunos tipos de estado necesitan campos adicionales o pueden cambiar el uso habitual de los campos comunes.

Cuando haya creado y ejecutado flujos de trabajo estándar, podrá acceder a la información acerca de cada estado, su entrada y su salida, cuándo se activó y durante cuánto tiempo estuvo activo

mediante la página Detalles de ejecución de la consola de Step Functions. Para obtener más información, consulte Visualización y depuración de ejecuciones en la consola de Step Functions.

Una vez que haya creado y ejecutado las ejecuciones de flujo de trabajo rápido y si está activado el registro en el flujo de trabajo rápido, podrá acceder a la información sobre la ejecución en Amazon CloudWatch Logs o en la consola de Step Functions. Para obtener más información, consulte Visualización y depuración de ejecuciones en la consola de Step Functions.

Temas

Lenguaje de estados de Amazon

Amazon States Language es un lenguaje estructurado basado en JSON que se utiliza para definir máquinas de estado: un conjunto de estados, que realizan tareas (estados `Task`), determinan a qué estados se debe pasar a continuación (estados `Choice`), detienen una ejecución con un error (estados `Fail`), etcétera.

Para obtener más información, consulte la Especificación de Amazon States Language y Statelint, una herramienta que valida el código de Amazon States Language.

Para crear una máquina de estados en la consola de Step Functions con Amazon States Language, consulte Introducción.

> **Note**
>
> Si define las máquinas de estado fuera de la consola de Step Functions, por ejemplo, en un editor de su elección, debe guardar las definiciones de las máquinas de estado con la extensión .asl.json.

Ejemplo de especificación de Amazon States Language

```
{
  "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
    }
  }
}
```

Temas

- Estructura de las máquinas de estado
- Funciones intrínsecas
- Campos de estado comunes

Estructura de las máquinas de estado

Las máquinas de estado se definen utilizando texto JSON que representa una estructura que contiene los siguientes campos.

`Comment` (Opcional)

Descripción en lenguaje natural de la máquina de estado.

`StartAt` (Obligatorio)

Cadena que debe coincidir exactamente (mayúsculas y minúsculas) con el nombre de uno de los objetos de estado.

`TimeoutSeconds` (Opcional)

Número máximo de segundos que puede tardar una ejecución de la máquina de estado. Si se ejecuta durante más tiempo que el especificado, se produce un error en la ejecución con un nombre de error `States.Timeout`.

`Version` (Opcional)

Versión de Amazon States Language que se utiliza en la máquina de estado (el valor predeterminado es "1.0").

`States` (Obligatorio)

Objeto que contiene un conjunto de estados separados por comas.

El campo `States` contiene estados.

```
{
    "State1" : {
    },

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

Las máquinas de estado se definen por los estados que contienen y la relación entre ellos.

A continuación, se muestra un ejemplo.

```
{
  "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
    }
  }
}
```

Cuando se inicia una ejecución de esta máquina de estado, el sistema comienza con el estado al que se hace referencia en el campo `StartAt` (`"HelloWorld"`). Si este estado tiene un campo `"End": true`, la ejecución se detiene y se devuelven los resultados. De lo contrario, el sistema busca un campo `"Next":` y continúa con ese estado "next". Este proceso se repite hasta que el sistema alcanza un estado terminal (un estado con `"Type": "Succeed"`, `"Type": "Fail"` o `"End": true`) o se produce un error del sistema en tiempo de ejecución.

Las siguientes reglas se aplican a los estados de una máquina de estado:

- Los estados pueden tener lugar en cualquier orden dentro del bloque delimitado; el orden en el que aparecen no afecta al orden en el que se ejecutan. El contenido de los estados determina este orden.
- En una máquina de estado, solo puede haber un estado designado como `start`; esta designación se hace a través del valor del campo `StartAt` de la estructura principal. Este estado es uno de los que se ejecuta primero cuando se inicia la ejecución.
- Los estados cuyo campo `End` tiene el valor `true`, se consideran estados `end` (o `terminal`). En función de la lógica de la máquina de estado (por ejemplo, si la máquina de estado tiene varias ramificaciones de ejecución), es posible que haya varios estados `end`.
- Si la máquina de estado se compone de un solo estado, este podría ser tanto el estado `start` como el estado `end`.

Funciones intrínsecas

Amazon States Language proporciona varias funciones intrínsecas, también conocidas como tipos intrínsecos, que ayudan a realizar operaciones básicas de procesamiento de datos sin usar un estado `Task`. Las funciones intrínsecas son componentes que tienen un aspecto similar a las funciones de los lenguajes de programación. Se pueden usar para ayudar a los constructores de cargas a procesar los datos que entran y salen del campo `Resource` de un estado `Task`.

En Amazon States Language, las funciones intrínsecas se agrupan en las siguientes categorías, según el tipo de tarea de procesamiento de datos que se desee realizar:

- Funciones intrínsecas para matrices
- Funciones intrínsecas para la codificación y decodificación de datos
- Función intrínseca para el cálculo del hash
- Funciones intrínsecas para la manipulación de datos de JSON

- Funciones intrínsecas para operaciones matemáticas
- Función intrínseca para la operación de cadena
- Función intrínseca para la generación de identificadores únicos
- Función intrínseca para una operación genérica

> **Note**
>
> - Para utilizar funciones intrínsecas, debe especificar `.$` en el valor clave de las definiciones de las máquinas de estado, tal y como se muestra en el ejemplo siguiente:
>
> ```
> "KeyId.$": "States.Array($.Id)"
> ```
>
> - Puede anidar hasta 10 funciones intrínsecas dentro de un campo en sus flujos de trabajo. En el siguiente ejemplo se muestra un campo denominado *myArn* que incluye nueve funciones intrínsecas anidadas:
>
> ```
> "myArn.$": "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**
>
> Si utiliza Step Functions en un entorno de desarrollo local, asegúrese de utilizar la versión 1.12.0 o posterior para poder incluir todas las funciones intrínsecas en sus flujos de trabajo.

Campos que admiten funciones intrínsecas

En la tabla siguiente se muestra qué campos admiten funciones intrínsecas para cada estado.

Campos que admiten funciones intrínsecas

State	Pass	Task	Choice	Wait	Succeed	Fail	Parallel	Map
InputPath								
Parameters	✓	✓					✓	✓
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
Variable								
<Operador de comparación>Path								
TimeoutSecondsPath								
HeartbeatSecondsPath								
Credentials		✓						

Funciones intrínsecas para matrices

Utilice las siguientes funciones intrínsecas para realizar manipulaciones de matrices.

States.Array

La función intrínseca `States.Array` acepta cero o más argumentos. El intérprete devuelve una matriz JSON que contiene los valores de los argumentos en el orden indicado. Por ejemplo, en el caso de la entrada siguiente:

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

Podría usar

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

Que devolvería el siguiente resultado:

```
“BuildId”: [123456]
```

States.ArrayPartition

Utilice la función intrínseca `States.ArrayPartition` para particionar una matriz grande. También puede utilizar esta función intrínseca para dividir en sectores los datos y, a continuación, enviar la carga en fragmentos de menor tamaño.

Esta función intrínseca toma dos argumentos. El primer argumento es una matriz, mientras que el segundo define el tamaño del fragmento. El intérprete divide la matriz de entrada en varias matrices del tamaño especificado por el tamaño del fragmento. La longitud del último fragmento de matriz puede ser menor que la longitud de los fragmentos de matriz anteriores si el número de elementos restantes de la matriz es menor que el tamaño del fragmento.

Validación de entrada

- Debe especificar una matriz como valor de entrada para el primer argumento de la función.
- Debe especificar un entero positivo distinto de cero para el segundo argumento, que representa el valor del tamaño del fragmento.

  Si especifica un valor no entero para el segundo argumento, Step Functions lo redondeará al entero más cercano.
- La matriz de entrada no puede superar el límite de tamaño de carga de Step Functions de 256 KB.

Por ejemplo, en el caso de la matriz de entrada siguiente:

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

Puede usar la función `States.ArrayPartition` para dividir la matriz en fragmentos de cuatro valores:

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

Lo que devolvería los siguientes fragmentos de matriz:

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

En el ejemplo anterior, la función `States.ArrayPartition` genera tres matrices. Cada una de las dos primeras matrices contiene cuatro valores, según lo definido por el tamaño del fragmento. Una tercera matriz contiene el valor restante y su tamaño es menor que el tamaño de fragmento definido.

`States.ArrayContains`

Utilice la función intrínseca `States.ArrayContains` para determinar si un valor específico está presente en una matriz. Por ejemplo, puede usar esta función para detectar si hubo un error en una iteración de estado `Map`.

Esta función intrínseca toma dos argumentos. El primer argumento es una matriz, mientras que el segundo argumento es el valor que se debe buscar dentro de la matriz.

Validación de entrada

- Debe especificar una matriz como valor de entrada para el primer argumento de la función.
- Debe especificar un objeto JSON válido como segundo argumento.
- La matriz de entrada no puede superar el límite de tamaño de carga de Step Functions de 256 KB.

Por ejemplo, en el caso de la matriz de entrada siguiente:

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

```
}
```

Puede usar la función `States.ArrayContains` para encontrar el valor `lookingFor` dentro de la `inputArray`:

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

Como el valor almacenado en `lookingFor` está incluido en`inputArray`, `States.ArrayContains` devuelve el siguiente resultado:

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

`States.ArrayRange`

Utilice la función intrínseca `States.ArrayRange` para crear una nueva matriz que contenga un rango específico de elementos. La nueva matriz puede contener hasta 1000 elementos.

Esta función toma tres argumentos. El primer argumento es el primer elemento de la nueva matriz, el segundo argumento es el elemento final de la nueva matriz y el tercer argumento es el valor del incremento entre los elementos de la nueva matriz.

Validación de entrada

- Debe especificar valores enteros para todos los argumentos.

  Si especifica un valor no entero para alguno de los argumentos, Step Functions lo redondeará al entero más cercano.
- Debe especificar un valor distinto de cero para el tercer argumento.
- La matriz recién generada no puede contener más de 1000 elementos.

Por ejemplo, el siguiente uso de la función `States.ArrayRange` creará una matriz con un primer valor de 1, un valor final de 9 y los valores entre el primer valor y el valor final aumentarán en dos por cada elemento:

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

Lo que devolvería la siguiente matriz:

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

`States.ArrayGetItem`

Esta función intrínseca devuelve el valor de un índice especificado. Esta función toma dos argumentos. El primer argumento es una matriz de valores y el segundo argumento es el índice de matriz del valor que se va a devolver.

Por ejemplo, utilice los siguientes valores `inputArray` y `index`:

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

A partir de estos valores, puede usar la función `States.ArrayGetItem` para devolver el valor de la posición `index` 5 dentro de la matriz:

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

En este ejemplo, `States.ArrayGetItem` devolvería el siguiente resultado:

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

`States.ArrayLength`

La función intrínseca `States.ArrayLength` devuelve la longitud de una matriz. Tiene un argumento, la matriz cuya longitud se devuelve.

Por ejemplo, en el caso de la matriz de entrada siguiente:

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

Se puede utilizar `States.ArrayLength` para devolver la longitud de`inputArray`:

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

En este ejemplo, `States.ArrayLength` devolvería el siguiente objeto JSON que representa la longitud de la matriz:

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

States.ArrayUnique

La función intrínseca `States.ArrayUnique` elimina los valores duplicados de una matriz y devuelve una matriz que contiene solo elementos únicos. Esta función toma una matriz, que puede estar desordenada, como único argumento.

Por ejemplo, la siguiente `inputArray` contiene una serie de valores duplicados:

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

Puede usar la función `States.ArrayUnique` y especificar la matriz de la que desea eliminar los valores duplicados:

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

La función `States.ArrayUnique` devolvería la siguiente matriz que contiene solo elementos únicos y eliminaría todos los valores duplicados:

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

Funciones intrínsecas para la codificación y decodificación de datos

Utilice las siguientes funciones intrínsecas para codificar o decodificar datos según el esquema de codificación Base64.

States.Base64Encode

Utilice la función intrínseca `States.Base64Encode` para codificar datos según el esquema de codificación MIME Base64. Puede utilizar esta función para pasar datos a otros servicios AWS sin utilizar una función AWS Lambda.

Esta función toma una cadena de datos de hasta 10 000 caracteres para codificarla como único argumento.

Por ejemplo, considere la siguiente cadena `input`:

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

Puede utilizar la función `States.Base64Encode` para codificar la cadena `input` como una cadena MIME Base64:

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

La función `States.Base64Encode` devuelve los siguientes datos codificados en la respuesta:

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

`States.Base64Decode`

Utilice la función intrínseca `States.Base64Decode` para descodificar datos según el esquema de descodificación MIME Base64. Puede usar esta función para pasar datos a otros servicios AWS sin usar una función de Lambda.

Esta función utiliza una cadena de datos codificada en Base64 de hasta 10 000 caracteres para descodificarla como único argumento.

Por ejemplo, en el caso de la entrada siguiente:

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

Puede usar la función `States.Base64Decode` para descodificar la cadena en base64 y convertirla en una cadena en lenguaje natural:

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

En respuesta, la `States.Base64Decode function` devolvería los siguientes datos descodificados:

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

Función intrínseca para el cálculo del hash

`States.Hash`

Utilice la función intrínseca `States.Hash` para calcular el valor de hash de una entrada determinada. Puede usar esta función para pasar datos a otros servicios AWS sin usar una función de Lambda.

Esta función toma dos argumentos. El primer argumento son los datos cuyo valor hash se desea calcular. El segundo argumento es el algoritmo de hash que se utilizará para realizar el cálculo de hash. Los datos que proporcione deben ser una cadena de objeto que contenga 10 000 caracteres o menos.

El algoritmo de hash que especifique puede ser cualquiera de los siguientes:

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

Por ejemplo, puede usar esta función para calcular el valor de hash de la cadena `Data` utilizando el `Algorithm` especificado:

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

Puede usar la función `States.Hash` para calcular el valor de hash:

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

La función `States.Hash` devuelve el siguiente valor de hash como respuesta:

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

Funciones intrínsecas para la manipulación de datos de JSON

Utilice estas funciones para realizar operaciones básicas de procesamiento de datos en objetos JSON.

`States.JsonMerge`

Utilice la función intrínseca `States.JsonMerge` para combinar dos objetos JSON en un único objeto. Esta función toma tres argumentos. Los dos primeros argumentos son los objetos JSON

que se desea combinar. El tercer argumento es un valor booleano de `false`. Este valor booleano determina si el modo de combinación profunda está activado.

Actualmente, Step Functions solo admite el modo de combinación superficial; por lo tanto, debe especificar el valor booleano como `false`. En el modo superficial, si existe la misma clave en ambos objetos JSON, la clave del último objeto anula la misma clave del primer objeto. Además, los objetos anidados dentro de un objeto JSON no se combinan cuando se utiliza la combinación superficial.

Por ejemplo, puede usar la función `States.JsonMerge` para combinar los siguientes objetos JSON que comparten la clave a.

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

Puede especificar los objetos JSON1 y JSON2 como entradas en la función `States.JsonMerge` para combinarlos:

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

Como resultado, `States.JsonMerge` devuelve el siguiente objeto JSON combinado. En el objeto JSON combinado `output`, la clave a del objeto `json2` reemplaza a la clave a del objeto `json1`. Además, el objeto anidado de la clave a del objeto `json1` se descarta porque el modo superficial no admite la combinación de objetos anidados.

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

States.StringToJson

La función `States.StringToJson` toma como único argumento una ruta de referencia a una cadena JSON con caracteres de escape.

El intérprete aplica un analizador JSON y devuelve el formulario JSON analizado de la entrada. Por ejemplo, puede utilizar esta función para incluir la siguiente cadena de entrada entre caracteres de escape:

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

Utilice la función `States.StringToJson` y especifique `escapedJsonString` como argumento de entrada:

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

La función `States.StringToJson` devuelve el siguiente resultado:

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

`States.JsonToString`

La función `States.JsonToString` solo toma un argumento, que es la ruta que contiene los datos JSON que se devuelven como una cadena sin caracteres de escape. El intérprete devuelve una cadena con texto JSON que representa los datos especificados en la ruta. Por ejemplo, puede proporcionar la siguiente ruta JSON que contiene un valor con caracteres de escape:

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

Proporcione a la función `States.JsonToString` los datos contenidos en `unescapedJson`:

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

La función `States.JsonToString` devuelve la siguiente respuesta:

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

Funciones intrínsecas para operaciones matemáticas

Utilice estas funciones para realizar operaciones matemáticas.

`States.MathRandom`

Utilice la función intrínseca `States.MathRandom` para devolver un número al azar entre el número inicial (incluido) y el número final (excluido) especificados.

Puede usar esta función para distribuir una tarea específica entre dos o más recursos.

Esta función toma tres argumentos. El primer argumento es el número inicial, el segundo argumento es el número final y el último argumento controla el valor de inicio. El argumento del valor de inicio es opcional. Si utiliza esta función con el mismo valor de inicio, devolverá un número idéntico.

> **Important**
>
> Como la función `States.MathRandom` no devuelve números aleatorios criptográficamente seguros, le recomendamos que no la utilice para aplicaciones sensibles a la seguridad.

Validación de entrada

- Debe especificar valores enteros para los argumentos del número inicial y del número final.

  Si especifica un valor no entero para el argumento del número inicial o del número final, Step Functions lo redondeará al entero más cercano.

Por ejemplo, para generar un número aleatorio entre uno y 999, puede usar los siguientes valores de entrada:

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

Para generar el número aleatorio, proporcione los valores `start` y `end` a la función `States.MathRandom`:

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

La función `States.MathRandom` devuelve el siguiente número aleatorio como respuesta:

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

`States.MathAdd`

Utilice la función intrínseca `States.MathAdd` para devolver la suma de dos números. Por ejemplo, puede usar esta función para incrementar los valores dentro de un bucle sin invocar una función de Lambda.

Validación de entrada

- Debe especificar valores enteros para todos los argumentos.

  Si especifica un valor no entero para uno o ambos argumentos, Step Functions lo redondeará al entero más cercano.
- Debe especificar valores enteros en el rango de -2147483648 a 2147483647.

Por ejemplo, puede utilizar los siguientes valores para restarle uno a 111:

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

A continuación, utilice la función `States.MathAdd` definiendo `value1` como valor inicial y `step` como valor para incrementar `value1` en:

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

La función `States.MathAdd` devolverá el siguiente número como respuesta:

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

Función intrínseca para la operación de cadena

`States.StringSplit`

Utilice la función intrínseca `States.StringSplit` para dividir una cadena en una lista de valores de cadena. Esta función toma dos argumentos. El primer argumento es una cadena y el segundo es el carácter delimitador que la función utilizará para dividir la cadena.

Example - Dividir una cadena de entrada utilizando un único carácter delimitador

Para este ejemplo, utilice `States.StringSplit` para dividir la siguiente `inputString`, que contiene una serie de valores separados por comas:

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

Utilice la función `States.StringSplit` y defina `inputString` como primer argumento y el carácter delimitador `splitter` como segundo argumento:

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

Como resultado, la función `States.StringSplit` devuelve la siguiente matriz de cadenas:

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

Example - Dividir una cadena de entrada con varios caracteres delimitadores

Para este ejemplo, utilice `States.StringSplit` para dividir la siguiente`inputString`, que contiene varios caracteres delimitadores:

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

Utilice la función `States.StringSplit` de la siguiente manera:

```
{
```

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

Como resultado, la función `States.StringSplit` devuelve la siguiente matriz de cadenas:

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

Función intrínseca para la generación de identificadores únicos

`States.UUID`

Utilice la función intrínseca `States.UUID` para devolver un identificador único universal (UUID v4) de la versión 4 generado con números al azar. Por ejemplo, puede usar esta función para llamar a otros servicios o recursos de AWS que necesiten un parámetro UUID o para insertar elementos en una tabla de DynamoDB.

Se llama a la función `States.UUID` sin especificar ningún argumento:

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

La función devuelve un UUID generado aleatoriamente, como en el siguiente ejemplo:

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

Función intrínseca para una operación genérica

`States.Format`

Utilice la función intrínseca `States.Format` para construir una cadena a partir de valores literales e interpolados. Esta función toma uno o más argumentos. El valor del primer argumento debe ser una cadena y puede incluir cero o más instancias de la secuencia de caracteres `{}`. Debe haber tantos argumentos restantes en la invocación de la función intrínseca como

ocurrencias de `{}`. El intérprete devuelve la cadena definida en el primer argumento, sustituyendo cada `{}` por el valor del argumento correspondiente a la posición en la invocación de función intrínseca.

Por ejemplo, puede usar las siguientes entradas de `name` de un individuo y una frase `template` en la que insertar su nombre:

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

Utilice la función `States.Format` y especifique la cadena `template` y la cadena que desee insertar en lugar de los caracteres `{}`:

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

o bien

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

Con cualquiera de las entradas anteriores, la función `States.Format` devuelve la cadena completa como respuesta:

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

Caracteres reservados en funciones intrínsecas

Los siguientes caracteres están reservados para funciones intrínsecas y deben incluirse en secuencias de escape con ('\') si desea que aparezcan en el valor: '{} y \.

Si el carácter \ debe aparecer como parte del valor sin servir como carácter de escape, debe incluirlo en una secuencia de escape con una barra invertida. Las siguientes secuencias de caracteres de escape se utilizan con funciones intrínsecas:

- La cadena literal `\'` representa`'`.
- La cadena literal `\{` representa`{`.
- La cadena literal `\}` representa`}`.

- La cadena literal \\ representa\.

En JSON, las barras invertidas contenidas en un valor literal de cadena se deben incluir en una secuencia de escape con otra barra invertida. La lista equivalente para JSON es:

- La cadena de escape \\\' representa \'.
- La cadena de escape \\\{ representa \{.
- La cadena de escape \\\} representa \}.
- La cadena de escape \\\\ representa \\.

> **Note**
>
> Si se encuentra una barra invertida de escape abierta \ en la cadena de invocación intrínseca, el intérprete devolverá un error de tiempo de ejecución.

Campos de estado comunes

`Type` (Obligatorio)

El tipo de estado.

`Next`

El nombre del siguiente estado que se ejecuta cuando finaliza el estado actual. Algunos tipos de estado, como `Choice`, permiten varios estados de transición.

Si el estado actual es el último estado de su flujo de trabajo o un estado terminal., como Succeed o Fail, no es necesario que especifique el campo `Next`.

`End`

Designa este estado como un estado terminal (termina la ejecución) si está establecido en `true`. Puede haber un número cualquiera de estados terminales por máquina de estado. Solo se puede utilizar `Next` o `End` en un estado. Algunos tipos de estado, como `Choice`, o estados terminales, como Succeed y Fail, no admiten ni utilizan el campo `End`.

`Comment` (opcional)

Almacena una descripción en lenguaje natural del estado.

`InputPath` (opcional)

Una ruta que selecciona una parte de la entrada del estado que se va a pasar a la tarea del estado para su procesamiento. Si se omite, tiene el valor de $, que designa toda la entrada. Para obtener más información, consulte Procesamiento de entrada y salida.

`OutputPath` (opcional)

Una ruta que selecciona una parte de la salida del estado que se va a pasar al estado siguiente. Si se omite, tiene el valor de $, que designa toda la salida. Para obtener más información, consulte Procesamiento de entrada y salida.

Pass

El estado `Pass` (`"Type": "Pass"`) pasa los datos de entrada a la salida sin realizar ninguna tarea. Los estados `Pass` son útiles cuando para crear y depurar máquinas de estado.

También puede usar un estado `Pass` para transformar la entrada de estado de JSON mediante filtros y, a continuación, pasar los datos transformados al siguiente estado de los flujos de trabajo. Para obtener información sobre la transformación de entradas, consulte InputPath, Parámetros y ResultSelector.

Además de los campos de estado comunes, los estados `Pass` admiten los siguientes campos.

`Result` (opcional)

Hace referencia al resultado de una tarea virtual que se pasa al siguiente estado. Si se incluye el campo `ResultPath` en la definición de la máquina de estado, se coloca `Result` según lo especificado por `ResultPath` y se pasa al siguiente estado.

`ResultPath` (opcional)

Especifica dónde colocar la salida (respecto a la entrada) de la tarea virtual especificada en `Result`. Además, la entrada se filtra según el contenido del campo `OutputPath` (si existe) antes de utilizarla como salida del estado. Para obtener más información, consulte Procesamiento de entrada y salida.

`Parameters` (opcional)

Crea una colección de pares de valores de clave que se pasarán como entrada. Puede especificar `Parameters` como un valor estático o seleccionarlo de la entrada mediante una ruta. Para obtener más información, consulte InputPath, Parámetros y ResultSelector.

Ejemplo del estado Pass

A continuación, se muestra un ejemplo de un estado `Pass` que inserta algunos datos fijos en la máquina de estado, posiblemente con fines de prueba.

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

Supongamos que la entrada a este estado es esta:

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

En ese caso, la salida sería esta.

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

Estado de la tarea

Un estado `Task` (`"Type": "Task"`) representa una única unidad de trabajo realizada por una máquina de estado. Una tarea realiza su trabajo mediante una actividad o AWS Lambda función, mediante la integración con otras compatibles Servicios de AWS o mediante la invocación de una API de terceros, como Stripe.

El Amazon States Language representa tareas estableciendo un tipo de estado en `Task` y proporcionando a la tarea el Nombre de recurso de Amazon (ARN) de la actividad, la función de

Lambda o el punto de conexión de la API. La siguiente definición de estado de tarea invoca una función de Lambda denominada *HelloFunction*.

```
"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
}
```

En este tema

Tipos de tareas

Step Functions es compatible con los siguientes tipos de tareas, que se pueden especificar en una definición de estado de tarea:

- Actividad
- Funciones de Lambda
- Un compatible Servicio de AWS
- Una tarea HTTP

Para especificar un tipo de tarea, proporcione su ARN en el campo `Resource` de una definición de estado de tarea. En el ejemplo siguiente se muestra la sintaxis del campo `Resource`. Todos los tipos de tareas, excepto el que invoca una API de terceros, utilizan la siguiente sintaxis. Para obtener información sobre la sintaxis de la tarea HTTP, consulte Llamada a API de terceros.

En la definición del estado de la tarea, sustituya el texto en cursiva de la siguiente sintaxis por la información específica del AWS recurso.

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

En la lista siguiente, se explican los componentes individuales de esta sintaxis:

- `partition` es la AWS Step Functions partición que se va a utilizar, por lo general. `aws`
- `service` indica la que Servicio de AWS se utiliza para ejecutar la tarea y puede tener uno de los siguientes valores:
  - `states` para una actividad.
  - `lambda` para una función de Lambda. Si se integra con otros Servicios de AWS, por ejemplo, Amazon SNS o Amazon DynamoDB, utilice o. `sns` `dynamodb`
- `region` es el código de AWS región en el que se ha creado la actividad o el estado de Step Functions, el tipo de máquina, la función Lambda o cualquier otro AWS recurso.
- `account` es el Cuenta de AWS ID con el que ha definido el recurso.
- `task_type` es el tipo de tarea que se va a ejecutar. Puede ser uno de los siguientes valores:
  - `activity`: una actividad.
  - `function`: una función de Lambda.
  - *`servicename`*: el nombre de un servicio conectado compatible (consulte Integraciones optimizadas para Step Functions).
- `name` es el nombre del recurso registrado (el nombre de la actividad, el nombre de la función de Lambda o la acción de la API de servicio).

> **Note**
>
> Step Functions no admite hacer referencia a los ARN entre particiones o regiones. Por ejemplo, `aws-cn` no puede invocar tareas en la partición `aws` y viceversa.

En las secciones siguientes, se proporcionan más detalles sobre cada tipo de tarea.

Actividad

Las actividades representan procesos o subprocesos de trabajo implementados y alojados por usted, que realizan una tarea específica. Solo son compatibles con los flujos de trabajo estándar, pero no con los flujos de trabajo rápidos.

Los ARN de `Resource` de actividad usan la siguiente sintaxis.

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

> **Note**
>
> Debe crear actividades con Step Functions (mediante una CreateActivityacción de API o la consola de Step Functions) antes de utilizarlas por primera vez.

Para obtener más información sobre la creación de una actividad y la implementación de procesos de trabajo, consulte Actividades.

Funciones de Lambda

Las tareas Lambda ejecutan una función mediante. AWS Lambda Para especificar una función de Lambda, utilice el ARN de la función de Lambda del campo `Resource`.

Según el tipo de integración (integración optimizada o integración del SDK de AWS) que utilice para especificar una función de Lambda, la sintaxis del campo `Resource` de la función de Lambda varía.

La siguiente sintaxis del campo `Resource` es un ejemplo de una integración optimizada con una función de Lambda.

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

La siguiente sintaxis de `Resource` campo es un ejemplo de una integración del AWS SDK con una función Lambda.

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

La siguiente definición del estado `Task` muestra un ejemplo de una integración optimizada con la función de Lambda denominada *`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"
}
```

Una vez completada la función Lambda especificada en el `Resource` campo, su salida se envía al estado identificado en el `Next` campo (» NextState «).

A: compatible Servicio de AWS

Cuando se hace referencia a un recurso conectado, Step Functions llama directamente a las acciones de la API de un servicio compatible. Especifique el servicio y la acción en el campo `Resource`.

Los ARN de `Resource` de los servicios conectados usan la siguiente sintaxis.

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

> **Note**
>
> Para crear una conexión síncrona a un recurso conectado, añada `.sync` a la entrada *`APIname`* en el ARN. Para obtener más información, consulte Trabajo con otros servicios.

Por ejemplo:

```
{
 "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
    }
  }
```

```
}
```

Campos de estado de tarea

Además de los campos de estado comunes, los estados `Task`, disponen de los siguientes campos:

`Resource` (Obligatorio)

Un URI, en concreto, un ARN que identifica de forma inequívoca la tarea específica que se va a ejecutar.

`Parameters` (Opcional)

Se utiliza para pasar información a las acciones de la API de los recursos conectados. Los parámetros pueden utilizar una combinación de JSON estático y JsonPath. Para obtener más información, consulte Cómo pasar parámetros a una API de servicio.

`Credentials` (Opcional)

Especifica un rol de destino que debe asumir la función de ejecución de la máquina de estado antes de invocar el `Resource` especificado. Como alternativa, también puede especificar un valor de JSONPath o una función intrínseca que se resuelva en un ARN de rol de IAM en tiempo de ejecución en función de la entrada de ejecución. Si especifica un valor de JSONPath, debe anteponerle la notación `$.`.

Para ver ejemplos del uso de este campo en el estado `Task`, consulte Ejemplos del campo Credentials de estado de tarea. Para ver un ejemplo del uso de este campo para acceder a un AWS recurso multicuenta desde su máquina de estados, consulteTutorial: Acceso a recursos multicuenta AWS.

> **Note**
>
> Este campo es compatible con los Tipos de tareas que utilizan funciones Lambda y un servicio compatible AWS.

`ResultPath` (Opcional)

Especifica dónde se colocan (en la entrada) los resultados de la ejecución de la tarea especificada en `Resource`. La entrada se filtra según el contenido del campo `OutputPath` (si existe) antes de utilizarla como salida del estado. Para obtener más información, consulte Procesamiento de entrada y salida.

`ResultSelector` (Opcional)

Pase un conjunto de pares clave-valor, donde los valores sean estáticos o se seleccionen del resultado. Para obtener más información, consulte ResultSelector.

`Retry` (Opcional)

Una matriz de objetos, denominados "reintentadores", que definen una política de reintentos si el estado encuentra errores en tiempo de ejecución. Para obtener más información, consulte Ejemplos de máquina de estado que usan Retry y Catch.

`Catch` (Opcional)

Una matriz de objetos, denominados "receptores", que definen un estado alternativo. Este estado se ejecuta si el estado encuentra errores en tiempo de ejecución y su política de reintentos está agotada o no se ha definido. Para obtener más información, consulte Estados alternativos.

`TimeoutSeconds` (Opcional)

Especifica el tiempo máximo que puede ejecutarse una actividad o tarea antes de que se agote el tiempo de espera y se produzca un error States.Timeout. El valor del tiempo de inactividad debe ser un número entero positivo, distinto de cero. El valor predeterminado es 99999999.

El tiempo de inactividad empieza a contar cuando se inicia la tarea, por ejemplo, cuando se registra `ActivityStarted` o `LambdaFunctionStarted` en el historial de eventos de ejecución. En las actividades, el tiempo empieza a contar cuando `GetActivityTask` recibe un token y se registra `ActivityStarted` en el historial de eventos de ejecución.

Cuando se inicia una tarea, Step Functions espera a que el proceso de trabajo de la tarea o actividad responda correctamente o no dentro del período `TimeoutSeconds` especificado. Si el proceso de trabajo de la tarea o actividad no responde dentro de este tiempo, Step Functions marca la ejecución del flujo de trabajo como fallida.

`TimeoutSecondsPath` (Opcional)

Si desea proporcionar un valor de tiempo de espera de forma dinámica a partir de la entrada de estado con una ruta de referencia, utilice `TimeoutSecondsPath`. Una vez resuelta, la ruta de referencia debe seleccionar campos cuyos valores sean enteros positivos.

> **Note**
>
> Un estado `Task` no puede incluir tanto `TimeoutSeconds` como `TimeoutSecondsPath`.

`HeartbeatSeconds` (Opcional)

Determina la frecuencia de las señales de latido que envía un proceso de trabajo de actividad durante la ejecución de una tarea. Los latidos indican que una tarea aún se está ejecutando y necesita más tiempo para completarse. Los latidos impiden que se agote el tiempo de espera de una actividad o tarea durante la duración de `TimeoutSeconds`.

`HeartbeatSeconds` debe ser un valor entero positivo, distinto de cero, inferior al valor del campo `TimeoutSeconds`. El valor predeterminado es 99999999. Si transcurre más tiempo que los segundos especificados entre los latidos de la tarea, el estado de la tarea produce el error States.Timeout.

En las actividades, el tiempo empieza a contar cuando `GetActivityTask` recibe un token y se registra `ActivityStarted` en el historial de eventos de ejecución.

`HeartbeatSecondsPath` (Opcional)

Si desea proporcionar un valor de latido de forma dinámica desde la entrada de estado mediante una ruta de referencia, utilice `HeartbeatSecondsPath`. Una vez resuelta, la ruta de referencia debe seleccionar campos cuyos valores sean enteros positivos.

> **Note**
>
> Un estado `Task` no puede incluir tanto `HeartbeatSeconds` como `HeartbeatSecondsPath`.

Un estado `Task` debe establecer el campo `End` en `true` si el estado finaliza su ejecución, o bien debe proporcionar un estado en el campo `Next` que se ejecuta cuando finaliza el estado `Task`.

Ejemplos de definición de estado de tarea

En los siguientes ejemplos se muestra cómo puede especificar la definición de estado de tarea en función de sus necesidades.

- Especificar tiempos de espera de estado de tarea e intervalos de latido
  - Ejemplo de notificación de latido y tiempo de espera estático
  - Ejemplo de notificación de latido y tiempo de espera dinámico
- Uso del campo Credentials
  - Especificación del ARN del rol de IAM codificado de forma rígida

- Especificar JSONPath como un ARN de rol de IAM
- Especificar una función intrínseca como ARN del rol de IAM

Tiempos de espera de estado de tarea e intervalos de latido

Es aconsejable definir un valor de tiempo de espera y un intervalo de latido para actividades de ejecución prolongada. Para hacer esto se puede especificar los valores de tiempo de espera y latido o configurarlos de forma dinámica.

Ejemplo de notificación de latido y tiempo de espera estático

Cuando se complete `HelloWorld`, se ejecutará el siguiente estado (que aquí se llama `NextState`).

Si esta tarea no puede completarse en un plazo de 300 segundos o no envía notificaciones de latido a intervalos de 60 segundos, la tarea se marca como `failed`.

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

Ejemplo de notificación de latido y tiempo de espera dinámico

En este ejemplo, cuando se complete el AWS Glue trabajo, se ejecutará el siguiente estado.

Si esta tarea no se completa dentro del intervalo establecido de manera dinámica por el trabajo AWS Glue , la tarea se marca como `failed`.

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

  "Next": "NextState"
}
```

Ejemplos del campo Credentials de estado de tarea

Especificación del ARN del rol de IAM codificado de forma rígida

En el siguiente ejemplo se especifica un rol de IAM de destino que debe asumir el rol de ejecución de una máquina de estado para acceder a una función de Lambda entre cuentas denominada `Echo`. En este ejemplo, el ARN del rol de destino se especifica como un valor codificado de forma rígida.

```
{
  "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
    }
  }
}
```

Especificar JSONPath como un ARN de rol de IAM

En el siguiente ejemplo se especifica un valor de JSONPath, que se resolverá en un ARN de rol de IAM en tiempo de ejecución.

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

```
}
```

Especificar una función intrínseca como ARN del rol de IAM

En el siguiente ejemplo, se utiliza la función intrínseca `States.Format`, que se convierte en un ARN de rol de IAM en tiempo de ejecución.

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

Actividades

Las actividades son una característica de AWS Step Functions que permite tener una tarea en la máquina de estado donde el trabajo lo realiza un proceso de trabajo que puede hospedarse en Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), dispositivos móviles… Básicamente, en cualquier lugar.

Información general

En AWS Step Functions, las actividades son una forma de asociar código que se ejecuta en algún lugar (conocido como un proceso de trabajo de actividad) con una tarea específica en una máquina de estado. Puede crear una actividad con la consola de Step Functions o llamando a `CreateActivity`. Proporciona un Nombre de recurso de Amazon (ARN) para su estado de tarea. Utilice este ARN para sondear el estado de la tarea correspondiente al trabajo en su proceso de trabajo de actividad.

> **Note**
>
> Las actividades no tienen control de versiones y se espera que sean compatibles con versiones anteriores. Si debe realizar un cambio incompatible con versiones anteriores en una actividad, cree una nueva actividad en Step Functions utilizando un nombre único.

Un proceso de trabajo de actividad puede ser una aplicación que se ejecuta en una instancia de Amazon EC2, una función de AWS Lambda, un dispositivo móvil… Cualquier aplicación que pueda realizar una conexión HTTP, alojada en cualquier lugar. Cuando Step Functions llega a un estado de tarea de actividad, el flujo de trabajo espera a que un proceso de trabajo de actividad realice un sondeo de una tarea. Un proceso de trabajo de actividad sondea Step Functions mediante `GetActivityTask` y envía el ARN de la actividad relacionada. `GetActivityTask` devuelve una respuesta que incluye `input` (una cadena de entrada JSON de la tarea) y un `taskToken` (un identificador único de la tarea). Después de que el proceso de trabajo de actividad completa su trabajo, puede proporcionar un informe de si la operación se ha realizado correcta o incorrectamente mediante `SendTaskSuccess` o `SendTaskFailure`. Estas dos llamadas usan el `taskToken` que proporciona `GetActivityTask` para asociar el resultado con esa tarea.

API relacionadas con las tareas de actividad

Step Functions proporciona API para crear y enumerar actividades, solicitar una tarea y administrar el flujo de su máquina de estado en función de los resultados de su proceso de trabajo.

A continuación se indican las API de Step Functions que están relacionadas con las actividades:

- `CreateActivity`
- `GetActivityTask`
- `ListActivities`
- `SendTaskFailure`
- `SendTaskHeartbeat`
- `SendTaskSuccess`

> **Note**
>
> El sondeo de las tareas de actividad con `GetActivityTask` pueden provocar latencia en algunas implementaciones. Consulte Evitar la latencia al sondear tareas de actividad.

Espera para que finalice una tarea de actividad

Establezca `TimeoutSeconds` en la definición de la tarea para configurar cuánto tiempo espera un estado. Para mantener la tarea activa y en espera, envíe periódicamente un latido de su proceso de trabajo de actividad mediante `SendTaskHeartbeat` dentro del tiempo configurado en `TimeoutSeconds`. Si configura un tiempo de espera largo y envía activamente un latido, una actividad en Step Functions puede esperar hasta un año a que se complete una ejecución.

Por ejemplo, si necesita un flujo de trabajo que espere el resultado de un proceso largo, haga lo siguiente:

1. Cree una actividad con la consola o mediante `CreateActivity`. Tome nota del ARN de actividad.
2. Haga referencia a ese ARN en un estado de tarea de actividad en el definición de la máquina de estado y establezca `TimeoutSeconds`.
3. Implemente un proceso de trabajo de actividad que sondee el trabajo mediante `GetActivityTask`, haciendo referencia a ese ARN de actividad.
4. Utilice `SendTaskHeartbeat` periódicamente dentro del tiempo establecido en `HeartbeatSeconds` en la definición de tarea de máquina de estado para evitar que se agote el tiempo de espera de la tarea.
5. Comience una ejecución de su máquina de estado.
6. Comience su proceso de trabajo de actividad.

La ejecución se pone en pausa en el estado de la tarea de actividad y espera a que el proceso de trabajo de actividad realice un sondeo de una tarea. Una vez que se proporciona un `taskToken` a su proceso de trabajo de actividad, su flujo de trabajo esperará a que `SendTaskSuccess` o `SendTaskFailure` proporcione un estado. Si la ejecución no recibe ninguno de ellos ni una llamada a `SendTaskHeartbeat` antes del tiempo configurado en `TimeoutSeconds`, se producirá un error en la ejecución y el historial de ejecución contendrá un evento `ExecutionTimedOut`.

Pasos siguientes

Para obtener información más detallada sobre la creación de máquinas de estado que utilizan trabajadores de una actividad, consulte:

- Crear una máquina de estado de actividad con Step Functions
- Ejemplo de proceso de trabajo de actividad en Ruby

Ejemplo de proceso de trabajo de actividad en Ruby

A continuación se muestra un ejemplo de un proceso de trabajo de actividad que utiliza AWS SDK for Ruby para mostrarle cómo utilizar las prácticas recomendadas e implementar su propio proceso de trabajo de actividad.

El código implementa un patrón consumidor-productor con un número configurable de subprocesos para sondeadores y procesos de trabajo de actividad. El subprocesos de sondeador están realizando constantemente un sondeo largo de la tarea de actividad. Cuando se recupera una tarea de actividad, se pasa a través de una cola de bloqueo limitada para que el subproceso de actividad la recoja.

- Para obtener más información sobre AWS SDK for Ruby, consulte la Referencia de la API de AWS SDK for Ruby.
- Para descargar este código y los recursos relacionados, consulte el repositorio step-functions-ruby-activity-worker en GitHub.

El siguiente código Ruby es el principal punto de entrada de este ejemplo de proceso de trabajo de actividad de Ruby.

```
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
```

El código incluye valores predeterminados que puede modificar para hacer referencia a su actividad, así como para adaptarlo a su implementación específica. Este código toma como entrada la lógica de implementación real, le permite incluir referencias a una actividad y unas credenciales específicas, y le permite configurar el número de subprocesos y el retraso de latido. Para obtener más información y descargar el código, consulte Step Functions Ruby Activity Worker.

Elemento	Descripción
`require_relative`	Ruta relativa al siguiente código de proceso de trabajo de actividad de ejemplo.
`region`	Región de AWS de la actividad.
`workers_count`	Número de subprocesos del proceso de trabajo de actividad. Para la mayoría de las implementaciones, deberían ser suficient es entre 10 y 20 subprocesos. Cuanto más tiempo tarde la actividad en procesarse, más subprocesos podría necesitar. Para realizar una estimación, multiplique el número de actividades de proceso por segundo por la latencia de procesamiento de actividades del percentil 99th, en segundos.
`pollers_count`	Número de subprocesos de los sondeador es. Deberían ser suficientes entre 10 y 20 subprocesos para la mayoría de las implement aciones.
`heartbeat_delay`	Retraso entre latidos, en segundos.

Elemento	Descripción
`input`	Lógica de implementación de la actividad.

A continuación se presenta el proceso de trabajo de actividad de Ruby, al que se hace referencia con `../lib/step_functions/activity` en su código.

```
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

El estado `Choice` (`"Type": "Choice"`) agrega una lógica condicional a la máquina de estado.

Además de la mayoría de los campos de estado comunes, los estados `Choice` contienen los campos adicionales que se indican a continuación.

`Choices` (Obligatorio)

Matriz de reglas de Choice que determina qué estado provoca que la máquina de estado adopte el siguiente estado. Para comparar una variable de entrada con un valor específico se utiliza un operador de comparación en una regla de Choice. Por ejemplo, al usar reglas Choice, puede comparar si una variable de entrada es mayor o menor que 100.

Cuando se ejecuta un estado `Choice`, evalúa cada regla Choice como verdadera o falsa. Según el resultado de esta evaluación, Step Functions pasa al siguiente estado del flujo de trabajo.

Debe definir al menos una regla en el estado `Choice`.

`Default` (opcional, recomendado)

Nombre del estado que se va a adoptar si no tiene lugar ninguna de las transiciones de `Choices`.

> **Important**
>
> Los estados `Choice` no admiten el campo `End`. Además, solamente utilizan `Next` dentro del campo `Choices`.

> **Tip**
>
> Para implementar un ejemplo de un flujo de trabajo en el que se utilice un estado `Choice` en su Cuenta de AWS, consulte el Módulo 5: Estado de elección y estado del mapa de El workshop de AWS Step Functions.

Reglas de Choice

Un estado de `Choice` debe tener un campo `Choices` cuyo valor sea una matriz no vacía. Cada elemento de esta matriz es un objeto llamado regla de Choice, que contiene lo siguiente:

- Una comparación: dos campos que especifican la variable de entrada que se va a comparar, el tipo de comparación y el valor con el que se va a comparar la variable de entrada. Las reglas de Choice permiten la comparación entre dos variables. Dentro de una regla de Choice, el valor de la variable se puede comparar con otro valor de la entrada de estado añadiendo `Path` al nombre de los operadores de comparación compatibles. Los valores de los campos `Variable` y Ruta de una comparación deben ser rutas de referencia válidas.
- Un campo **`Next`**: el valor de este campo debe coincidir con el nombre del estado en que se encuentra la máquina de estado.

En el ejemplo siguiente, se comprueba si el valor numérico es igual a 1.

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

```
}
```

En el ejemplo siguiente, se comprueba si la cadena es igual a `MyString`.

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

En el ejemplo siguiente, se comprueba si la cadena es mayor que `MyStringABC`.

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

En el ejemplo siguiente, se comprueba si la cadena es nula.

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

En el ejemplo siguiente, se muestra cómo la regla StringEquals solo se evalúa cuando existe `$.keyThatMightNotExist` debido a la regla de Choice `IsPresent` anterior.

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

En el ejemplo siguiente, se comprueba si un patrón coincide con un comodín.

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

En el ejemplo siguiente, se comprueba si la marca de tiempo es igual a `2001-01-01T12:00:00Z`.

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

En el ejemplo siguiente, se compara una variable con otro valor de la entrada de estado.

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

Step Functions examina cada una de las reglas de Choice en el orden en el que aparecen en el campo `Choices`. A continuación, adopta el estado especificado en el campo `Next` de la primera regla de Choice en la que la variable coincide con el valor que corresponde al operador de comparación.

Se admiten los siguientes operadores de comparación:

- `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`

En cada uno de estos operadores, el valor correspondiente debe ser del tipo adecuado: cadena, número, booleano o marca de tiempo. Step Functions no intenta asociar un campo numérico con un valor de cadena. Sin embargo, como los campos de marca de tiempo son cadenas, es posible que un campo que se considera una marca de tiempo cumpla una condición establecida con un comparador `StringEquals`.

> **Note**
>
> Por motivos de interoperabilidad, no presuponga que las comparaciones numéricas funcionan con valores que están fuera del rango de magnitud o precisión que el tipo de datos binary64 de IEEE 754-2008 representa. En particular, es posible que los enteros que están fuera del rango $[-2^{53}+1, 2^{53}-1]$ no puedan compararse del modo esperado.
> Las marcas de tiempo (por ejemplo, `2016-08-18T17:33:00Z`) deben ajustarse al perfil RFC3339 de ISO 8601, con restricciones adicionales:
>
> - Las partes de fecha y hora deben separarse con una letra T mayúscula.

- Debe utilizarse una letra Z mayúscula para indicar que no se aplica ningún ajuste numérico de zona horaria.

Para comprender el comportamiento de comparación de cadenas, consulte la documentación de Java `compareTo`.
Los valores de los operadores `And` y `Or` deben ser matrices de reglas de Choice que no estén vacías y que no contengan campos `Next`. Del mismo modo, el valor de un operador `Not` tiene que ser una única regla de Choice que no debe contener campos `Next`.
Puede crear reglas de Choice anidadas y complejas utilizando `And`, `Not` y `Or`. Sin embargo, el campo `Next` solamente puede aparecer en las reglas de Choice de nivel superior.
La comparación de cadenas con patrones con uno o más caracteres comodín ("*") se puede realizar con el operador de comparación StringMatches. Se aplica escape al carácter comodín de manera estándar con `\\ (Ex: "\\*")`. Ningún otro carácter que no sea "*" tiene un significado especial para la coincidencia.

Ejemplo del estado Choice

A continuación, se muestra un ejemplo de un estado `Choice` y otros estados que adopta.

Note

Debe especificar el campo `$.type`. Si los datos de entrada del estado no contienen el campo `$.type`, la ejecución no se realizará correctamente y aparecerá un error en el historial de ejecución. En el campo `StringEquals` solo se puede especificar una cadena que coincida con un valor literal. Por ejemplo, `"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!"
```

```
}
```

En este ejemplo, la máquina de estado comienza con el siguiente valor de entrada.

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

Step Functions adopta el estado `ValueInTwenties` con arreglo al campo `value`.

Si no hay ninguna correspondencia con el campo `Choices` del estado `Choice`, se ejecuta el estado proporcionado en el campo `Default`. Si no es especifica el estado `Default`, se produce un error en la ejecución.

Wait

Un estado `Wait` (`"Type": "Wait"`) retrasa la máquina de estado durante el tiempo especificado. Se puede seleccionar un tiempo relativo, especificado en segundos desde el inicio del estado, o un tiempo de finalización absoluto, especificado como una marca de tiempo.

Además de los campos de estado comunes, los estados `Wait` tienen uno de los campos siguientes.

`Seconds`

Tiempo, en segundos, que se va a esperar hasta que comience el estado especificado en el campo `Next`. Debe especificar el tiempo como un valor entero positivo comprendido entre 0 y 99999999.

`Timestamp`

Tiempo absoluto que se va a esperar hasta que comience el estado especificado en el campo `Next`.

Las marcas de tiempo deben ajustarse al perfil RFC3339 de ISO 8601, con la restricción adicional de que las partes de fecha y hora deben separarse con una letra `T` mayúscula y debe usarse una letra `Z` mayúscula para dar cuenta de que no se aplica ningún ajuste numérico de la zona horaria; por ejemplo, `2024-08-18T17:33:00Z`.

> **Note**
>
> Actualmente, si se especifica el tiempo de espera como una marca de tiempo, Step Functions tiene en consideración el valor del tiempo hasta los segundos y trunca los milisegundos.

`SecondsPath`

Tiempo, en segundos, que se va a esperar hasta que comience el estado especificado en el campo `Next`. Se define mediante una ruta en los datos de entrada del estado.

Debe especificar un valor entero para este campo.

`TimestampPath`

Tiempo absoluto que se va a esperar hasta que comience el estado especificado en el campo `Next`. Se define utilizando una ruta en los datos de entrada del estado.

> **Note**
>
> Solo debe especificar uno: `Seconds`, `Timestamp`, `SecondsPath` o `TimestampPath`. Además, el tiempo de espera máximo que se puede especificar para los flujos de trabajo estándar y rápidos es de un año y cinco minutos, respectivamente.

Ejemplos del estado Wait

El estado `Wait` siguiente introduce un retraso de 10 segundos en una máquina de estado.

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

En el ejemplo siguiente, el estado `Wait` espera hasta un momento especificado en términos absolutos: 14 de marzo de 2024, a las 13:59 h UTC.

```
"wait_until" : {
```

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

No es necesario codificar de forma rígida la duración de Wait. Por ejemplo, en el caso de los datos de entrada siguientes:

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

Puede seleccionar el valor de "expirydate" de los datos de entrada utilizando una ruta de referencia.

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

Succeed

Un estado `Succeed` (`"Type": "Succeed"`) detiene correctamente una ejecución. El estado `Succeed` es un destino útil para las ramificaciones del estado `Choice` que no hacen otra cosa que detener la ejecución.

Dado que los estados `Succeed` son estados de terminal, no tienen ningún campo `Next` y no necesitan un campo `End`, como se muestra en el siguiente ejemplo.

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

Fail

Un estado `Fail` (`"Type": "Fail"`) detiene la ejecución de la máquina de estado y la marca como errónea, a menos que lo detecte un bloque `Catch`.

El estado `Fail` solo permite el uso de los campos `Type` y `Comment` del conjunto de campos de estado comunes. Además, el estado permite `Fail` los siguientes campos.

`Cause` (opcional)

Cadena personalizada que describe la causa del error. Puede especificar este campo con fines operativos o de diagnóstico.

`CausePath` (opcional)

Si desea proporcionar una descripción detallada de la causa del error de forma dinámica a partir de la entrada de estado utilizando una ruta de referencia, utilice `CausePath`. Una vez resuelto, la ruta de referencia debe seleccionar un campo que contenga un valor de cadena.

También puede especificar `CausePath` mediante una función intrínseca que devuelva una cadena. Estos elementos intrínsecos son: States.Format, States.JsonToString, States.ArrayGetItem, States.Base64Encode, States.Base64Decode, States.Hash y States.UUID.

> **Important**
>
> - Puede especificar `Cause` o `CausePath`, pero no ambos, en su definición de estado Fail.
> - Como práctica recomendada de seguridad de la información, le aconsejamos que elimine de la descripción de la causa toda la información confidencial y los detalles del sistema interno.

`Error` (opcional)

Un nombre de error que pueda proporcionar para gestionar los errores mediante los campos Retry o Catch. También puede proporcionar un nombre de error con fines operativos o de diagnóstico.

`ErrorPath` (opcional)

Si desea proporcionar un nombre para el error de forma dinámica a partir de la entrada de estado utilizando una ruta de referencia, utilice `ErrorPath`. Una vez resuelto, la ruta de referencia debe seleccionar un campo que contenga un valor de cadena.

También puede especificar `ErrorPath` mediante una función intrínseca que devuelva una cadena. Estos elementos intrínsecos son: States.Format, States.JsonToString, States.ArrayGetItem, States.Base64Encode, States.Base64Decode, States.Hash y States.UUID.

> **Important**
>
> - Puede especificar `Error` o `ErrorPath`, pero no ambos, en su definición de estado Fail.
> - Como práctica recomendada de seguridad de la información, le aconsejamos que elimine del nombre del error toda la información confidencial y los detalles del sistema interno.

Como el estado `Fail` siempre cierra la máquina de estado, no existe un campo `Next` ni se requiere un campo `End`.

Ejemplos de definición de estado Fail

En el siguiente ejemplo de definición de estado Fail se especifican valores de campo `Error` y `Cause` estáticos.

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

En el siguiente ejemplo de definición de estado Fail, se utilizan rutas de referencia de forma dinámica para resolver los valores de campo de `Error` y `Cause`.

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

En el siguiente ejemplo de definición de estado Fail, se utiliza la función intrínseca States.Format para especificar los valores de campo `Error` y `Cause` campo de forma dinámica.

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

```
  "ErrorPath": "States.Format('{}', $.Error)"
}
```

Parallel

El estado `Parallel` (`"Type": "Parallel"`) puede utilizarse para añadir ramificaciones de ejecución separadas en la máquina de estado.

Además de los campos de estado comunes, los estados `Parallel` incluyen los campos adicionales que se detallan a continuación.

`Branches` (Obligatorio)

Matriz de objetos que especifica las máquinas de estado que se van a ejecutar en paralelo. Cada uno de estos objetos de máquina de estado debe tener los campos `States` y `StartAt`, cuyos significados son exactamente iguales a los del nivel superior de una máquina de estado.

`ResultPath` (Opcional)

Especifica en qué lugar (de los datos de entrada) se va a situar la salida de las ramificaciones. La entrada se filtra según el contenido del campo `OutputPath` (si existe) antes de utilizarla como salida del estado. Para obtener más información, consulte Procesamiento de entrada y salida.

`ResultSelector` (Opcional)

Pase un conjunto de pares clave-valor, donde los valores sean estáticos o se seleccionen del resultado. Para obtener más información, consulte ResultSelector.

`Retry` (Opcional)

Una matriz de objetos, llamados "reintentadores", que definen una política de reintentos en caso de que el estado encuentre errores en tiempo de ejecución. Para obtener más información, consulte Ejemplos de máquina de estado que usan Retry y Catch.

`Catch` (Opcional)

Una matriz de objetos, denominados "receptores", que definen un estado alternativo que se ejecuta si el estado encuentra errores en tiempo de ejecución y su política de reintentos está agotada o no se ha definido. Para obtener más información, consulte Estados alternativos.

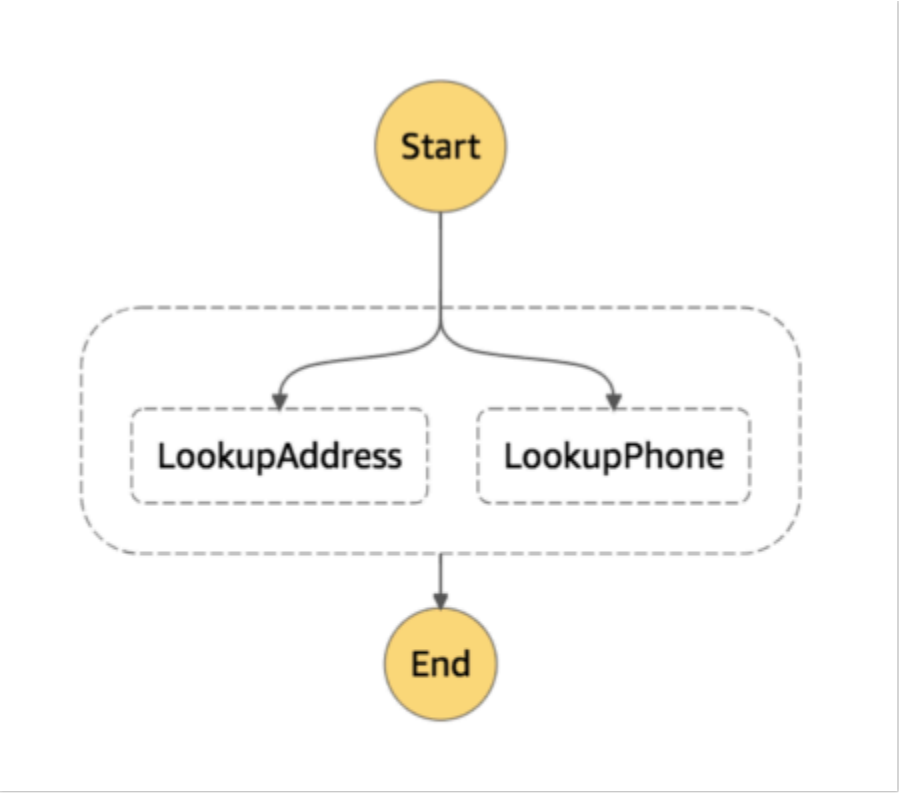
Un `Parallel` estado hace AWS Step Functions que cada rama se ejecute, empezando por el estado nombrado en el `StartAt` campo de esa rama, de la forma más simultánea posible, y que

espere hasta que todas las ramas terminen (alcancen un estado terminal) antes de procesar el `Next` campo del `Parallel` estado.

Ejemplo del estado Parallel

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

En este ejemplo, las ramificaciones `LookupAddress` y `LookupPhone` se ejecutan en paralelo. A continuación, se muestra el aspecto del flujo de trabajo visual en la consola de Step Functions.



Cada una de las ramificaciones debe ser autónoma. Los estados de una ramificación de un estado `Parallel` no deben incluir un campo `Next` que tenga como destino un campo que está fuera de la ramificación ni ningún otro estado fuera de la ramificación puede cambiar a esa ramificación.

Procesamiento de entrada y salida del estado Parallel

El estado `Parallel` proporciona a cada ramificación una copia de sus propios datos de entrada (sujetos a las modificaciones realizadas por el campo `InputPath`). Genera una salida que es una matriz con un elemento para cada ramificación, que contiene la salida de dicha ramificación. No es necesario que todos los elementos sean del mismo tipo. La matriz de salida puede insertarse en los datos de entrada (y el conjunto enviado como salida del estado `Parallel`) a través del campo `ResultPath` de la forma habitual (consulte Procesamiento de entrada y salida).

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

Si al estado `FunWithMath` se le proporcionó como entrada la matriz `[3, 2]`, tanto el estado `Add` como el estado `Subtract` recibirán esa matriz como datos de entrada. El resultado de las tareas `Add` y `Subtract` sería la suma y la diferencia entre los elementos de la matriz 3 y 2, es decir, 5 y 1, mientras que la salida del estado `Parallel` sería una matriz.

```
[ 5, 1 ]
```

> **Tip**
>
> Si el estado Parallel o Map que utiliza en sus máquinas de estado devuelve una matriz de matrices, puede transformarlas en una matriz plana con el campo ResultSelector. Para obtener más información, consulte Aplanamiento de una matriz de matrices.

Control de errores

Si alguna ramificación no se ejecuta correctamente debido a un error no controlado o porque cambia a un estado `Fail`, se considera que el error afecta a todo el estado `Parallel` y se detienen todas las ramificaciones. Si el propio estado `Parallel` no controla el error, Step Functions detendrá la ejecución con un error.

> **Note**
>
> Cuando un estado Parallel no se ejecuta correctamente, las funciones de Lambda invocadas se siguen ejecutando y los procesos de trabajo de actividad que procesan un token de tarea no se detienen.
>
> - Para detener actividades de ejecución prolongada, utilice latidos para detectar si Step Functions ha detenido su ramificación y detener los procesos de trabajo que están procesando tareas. Llamar a `SendTaskHeartbeat`, `SendTaskSuccess` o `SendTaskFailure` generará un error si el estado ha producido un error. Consulte Errores de latido.
> - La ejecución de funciones de Lambda no se puede detener. Si ha implementado una alternativa, utilice un estado `Wait` de modo que el trabajo de limpieza se realice una vez finalizada la función de Lambda.

Map

Utilice el estado `Map` para ejecutar un conjunto de pasos de flujo de trabajo para cada elemento de un conjunto de datos. Las iteraciones del estado `Map` se ejecutan en paralelo, lo que permite procesar un conjunto de datos rápidamente. Los estados `Map` pueden usar varios tipos de entrada, como una matriz JSON, una lista de objetos de Amazon S3 o un archivo CSV.

Step Functions proporciona dos tipos de modos de procesamiento para usar el estado `Map` en sus flujos de trabajo: modo En línea y modo Distribuido.

Para obtener información acerca de estos modos y de cómo se utiliza el estado `Map` en cualquiera de ellos, consulte los siguientes temas:

- Modos de procesamiento del estado Map
  - Uso del estado Map en modo En línea
  - Uso del estado Map en modo distribuido

> **Tip**
>
> Para implementar un ejemplo de un flujo de trabajo en el que se utiliza un estado `Map` en su Cuenta de AWS, consulte el Módulo 5: Estado Choice y estado Map de El workshop de AWS Step Functions.

Modos de procesamiento del estado Map

Step Functions proporciona los siguientes modos de procesamiento para el estado `Map`, según cómo desee procesar los elementos de un conjunto de datos.

- En línea: modo de simultaneidad limitada. En este modo, cada iteración del estado `Map` se ejecuta en el contexto del flujo de trabajo que contiene el estado `Map`. Step Functions añade el historial de ejecución de estas iteraciones al historial de ejecución del flujo de trabajo principal. De forma predeterminada, los estados `Map` se ejecutan en modo En línea.

  En este modo, el estado `Map` solo acepta una matriz JSON como entrada. Además, este modo admite hasta 40 iteraciones simultáneas.

  Para obtener más información, consulte Uso del estado Map en modo En línea.
- Distribuido: modo de alta simultaneidad. En este modo, el estado `Map` ejecuta cada iteración como una ejecución de flujo de trabajo secundario, lo que permite una alta simultaneidad de hasta 10 000 ejecuciones de flujos de trabajo secundarios en paralelo. Cada ejecución de flujo de trabajo secundario tiene su propio historial de ejecución independiente del flujo de trabajo principal.

  En este modo, el estado `Map` puede aceptar como entrada una matriz JSON o un origen de datos de Amazon S3, como un archivo CSV.

Para obtener más información, consulte Uso del estado Map en modo distribuido.

El modo que debe utilizar depende de cómo desee procesar los elementos de un conjunto de datos. Utilice el estado `Map` en modo En línea si el historial de ejecución del flujo de trabajo no va a superar las 25 000 entradas o si no necesita más de 40 iteraciones simultáneas.

Utilice el estado `Map` en modo Distribuido cuando necesite orquestar cargas de trabajo paralelas a gran escala que cumplan cualquier combinación de las siguientes condiciones:

- El tamaño del conjunto de datos supera los 256 KB.
- El historial de eventos de ejecución del flujo de trabajo supera las 25 000 entradas.
- Necesita una simultaneidad de más de 40 iteraciones paralelas.

Temas

Diferencias entre el modo En línea y el modo Distribuido

En la siguiente tabla, se muestran las diferencias entre los modos En línea y Distribuido.

Modo En línea	Modo Distribuido
Supported data sources	
Acepta como entrada una matriz JSON transferida desde un paso anterior del flujo de trabajo.	Acepta los siguientes orígenes de datos como entrada: • Matriz JSON transferida desde un paso anterior del flujo de trabajo • Archivo JSON en un bucket de Amazon S3 que contiene una matriz • Archivo CSV en un bucket de Amazon S3 • Lista de objetos de Amazon S3

Modo En línea	Modo Distribuido
	• Inventario de Amazon S3
Map iterations	
En este modo, cada iteración del estado `Map` se ejecuta en el contexto del flujo de trabajo que contiene el estado `Map`. Step Functions añade el historial de ejecución de estas iteraciones al historial de ejecución del flujo de trabajo principal.	En este modo, el estado `Map` ejecuta cada iteración como una ejecución de flujo de trabajo secundario, lo que permite una alta simultaneidad de hasta 10 000 ejecuciones de flujos de trabajo secundarios en paralelo. Cada ejecución de flujo de trabajo secundario tiene su propio historial de ejecución independiente del flujo de trabajo principal.
Maximum concurrency for parallel iterations	
Permite ejecutar hasta 40 iteraciones con la máxima simultaneidad posible.	Permite realizar hasta 10 000 ejecuciones de flujos de trabajo secundarios en paralelo para procesar millones de elementos de datos a la vez.
Input payload and event history sizes	
Aplica un límite de 256 KB al tamaño de la carga de entrada y de 25 000 entradas al historial de eventos de ejecución.	Permite superar la limitación del tamaño de la carga, ya que el estado `Map` puede leer la entrada directamente de orígenes de datos de Amazon S3. En este modo, también puede superar las limitaciones del historial de ejecución, ya que las ejecuciones des flujo de trabajo secundarios iniciadas por el estado `Map` mantienen sus propios historiales de ejecución independientes del historial de ejecución del flujo de trabajo principal.
Monitoring and observability	

Modo En línea	Modo Distribuido
Puede revisar el historial de ejecución del flujo de trabajo desde la consola o invocando la acción `GetExecutionHistory` de la API. También puede ver el historial de ejecución a través de CloudWatch y X-Ray.	Cuando se ejecuta un estado `Map` en modo distribuido, Step Functions crea un recurso Map Run. Un Map Run hace referencia a un conjunto de ejecuciones de flujos de trabajo secundarios que inicia un estado Map Distribut ed. Puede ver un Map Run en la consola de Step Functions. También puede invocar la acción de la API `DescribeMapRun`. Un Map Run también emite métricas a CloudWatch. Para obtener más información, consulte Examen de Map Run de un estado Map Distributed.

Uso del estado Map en modo En línea

De forma predeterminada, los estados `Map` se ejecutan en modo En línea. En el modo En línea, el estado Map solo acepta una matriz JSON como entrada. Recibe esta matriz de un paso anterior del flujo de trabajo. En este modo, cada iteración del estado `Map` se ejecuta en el contexto del flujo de trabajo que contiene el estado `Map`. Step Functions añade el historial de ejecución de estas iteraciones al historial de ejecución del flujo de trabajo principal.

En este modo, el estado `Map` admite hasta 40 iteraciones simultáneas.

Un estado `Map` configurado como en línea se conoce como estado Map en línea. Utilice el estado `Map` en modo En línea si el historial de ejecución del flujo de trabajo no va a superar las 25 000 entradas o si no necesita más de 40 iteraciones simultáneas.

Para ver una introducción al uso del estado Map en línea, consulte el tutorial Repetir una acción utilizando el estado Inline Map.

Contenido

- Ejemplo de estado Map en línea
- Ejemplo de estado Map en línea con ItemSelector
- Procesamiento de entrada y salida del estado Map en línea

Conceptos clave de este tema

Modo En línea

Un modo de simultaneidad limitada del estado `Map`. En este modo, cada iteración del estado `Map` se ejecuta en el contexto del flujo de trabajo que contiene el estado `Map`. Step Functions añade el historial de ejecución de estas iteraciones al historial de ejecución del flujo de trabajo principal. Los estados `Map` se ejecutan de manera predeterminada en el modo En línea.

Este modo solo acepta una matriz JSON como entrada y admite hasta 40 iteraciones simultáneas.

Estado Map en línea

Un estado `Map` configurado en el modo En línea.

Flujo de trabajo de Map

El conjunto de pasos que ejecuta el estado `Map` para cada iteración.

Iteración del estado Map

Una repetición del flujo de trabajo definido dentro del estado `Map`.

Campos del estado Map en línea

Para usar el estado Map en línea en los flujos de trabajo, especifique uno o más de estos campos. Estos campos se especifican además de los campos de estado comunes.

`Type` (Obligatorio)

Establece el tipo de estado, por ejemplo `Map`.

`ItemProcessor` (Obligatorio)

Contiene los siguientes objetos JSON que especifican el modo y la definición de procesamiento del estado `Map`.

La definición contiene el conjunto de pasos que se deben repetir para procesar cada elemento de la matriz.

- `ProcessorConfig` – Un objeto JSON opcional que especifica el modo de procesamiento del estado `Map`. Este objeto contiene el subcampo `Mode`. El valor predeterminado de este campo es `INLINE`, que usa el estado `Map` en el modo En línea.

  En este modo, si se produce un error en cualquier iteración, se produce un error en el estado `Map`. Todas las iteraciones se detienen cuando el estado `Map` produce un error.

- `StartAt`— Especifica una cadena que indica el primer estado de un flujo de trabajo. Esta cadena debe coincidir exactamente (mayúsculas y minúsculas) con el nombre de uno de los objetos de estado. Este estado se ejecuta primero para cada elemento del conjunto de datos. Cualquier entrada de ejecución que se proporcione al estado `Map` pasará primero al estado `StartAt`.

- `States` – Objeto JSON que contiene un conjunto de estados delimitados por comas. En este objeto, se define el Map workflow.

  > **Note**
  >
  > - Los estados del campo `ItemProcessor` solo pueden hacer la transición entre sí. Ningún estado fuera del campo `ItemProcessor` puede pasar a un estado dentro de él.
  > - El campo `ItemProcessor` reemplaza al campo `Iterator`, ahora obsoleto. Aunque puede seguir incluyendo estados `Map` que usen el campo `Iterator`, le recomendamos encarecidamente que sustituya este campo por `ItemProcessor`.
  >
  >   Actualmente, Step Functions Local no es compatible con el campo `ItemProcessor`. Le recomendamos que utilice el campo `Iterator` con Step Functions Local.

`ItemsPath` (opcional)

Especifica una ruta de referencia mediante la sintaxis JsonPath. Esta ruta selecciona el nodo JSON que contiene la matriz de elementos dentro de la entrada de estado. Para obtener más información, consulte ItemsPath.

`ItemSelector` (opcional)

Anula los valores de los elementos de la matriz de entrada antes de pasarlos a cada iteración del estado `Map`.

En este campo, se especifica un JSON válido que contiene una colección de pares clave-valor. Estos pares pueden contener lo siguiente:

- Valores estáticos se definen en la definición de la máquina de estado.
- Los valores se seleccionan de la entrada de estado mediante una ruta.
- Valores a los que se accede desde el objeto de contexto.

Para obtener más información, consulte ItemSelector.

El campo `ItemSelector` reemplaza al campo `Parameters`, ahora obsoleto. Aunque puede seguir incluyendo estados `Map` que usen el campo `Parameters`, le recomendamos encarecidamente que sustituya este campo por `ItemSelector`.

`MaxConcurrency` (opcional)

Especifica un valor entero que proporciona el límite superior del número de iteraciones del estado `Map` que se pueden ejecutar en paralelo. Por ejemplo, un valor de `MaxConcurrency` de 10 limitará el estado `Map` a 10 iteraciones que se ejecuten simultáneamente.

> **Note**
>
> Las iteraciones simultáneas pueden estar limitadas. Cuando ocurra esto, algunas iteraciones no comenzarán hasta que se completen las anteriores. La probabilidad de que esto ocurra aumenta cuando la matriz de entrada tiene más de 40 elementos.
> Para lograr una mayor simultaneidad, considere Uso del estado Map en modo distribuido.

El valor predeterminado es `0`, que no limita la simultaneidad. Step Functions invoca las iteraciones de la forma más simultánea posible.

Un valor de `MaxConcurrency` de `1` invoca a `ItemProcessor` una vez para cada elemento de la matriz. Los elementos de la matriz se procesan en el orden en que aparecen en la entrada. Step Functions no inicia una nueva iteración hasta que se completa la iteración anterior.

`MaxConcurrencyPath` (opcional)

Si desea proporcionar un valor máximo de simultaneidad de forma dinámica a partir de la entrada de estado mediante una ruta de referencia, utilice `MaxConcurrencyPath`. Una vez resuelta, la ruta de referencia debe seleccionar un campo cuyo valor sea un entero no negativo.

> **Note**
>
> Un estado `Map` no puede incluir tanto `MaxConcurrency` como `MaxConcurrencyPath`.

`ResultPath` (opcional)

Especifica en qué parte de la entrada se va a almacenar la salida de las iteraciones del estado `Map`. A continuación, el estado Map filtra la entrada según lo especificado por el campo `OutputPath`, si se especifica. A continuación, utiliza la entrada filtrada como salida del estado. Para obtener más información, consulte Procesamiento de entrada y salida.

`ResultSelector` (opcional)

Pase una colección de pares clave-valor, donde los valores sean estáticos o se seleccionen del resultado. Para obtener más información, consulte ResultSelector.

> **Tip**
>
> Si el estado Parallel o Map que utiliza en sus máquinas de estado devuelve una matriz de matrices, puede transformarlas en una matriz plana con el campo ResultSelector. Para obtener más información, consulte Aplanamiento de una matriz de matrices.

`Retry` (opcional)

Una matriz de objetos, denominados "reintentadores", que definen una política de reintentos. Los estados utilizan una política de reintentos cuando encuentran errores en tiempo de ejecución. Para obtener más información, consulte Ejemplos de máquina de estado que usan Retry y Catch.

> **Note**
>
> Si define "reintentadores" para el estado Map en línea, la política de reintentos se aplicará a todas las iteraciones del estado `Map` y no solo a las iteraciones con errores. Por ejemplo, el estado `Map` contiene dos iteraciones correctas y una iteración con error. Si ha definido el campo `Retry` para el estado `Map`, la política de reintentos se aplicará a las tres iteraciones del estado `Map` y no solo a la iteración con error.

`Catch` (opcional)

Una matriz de objetos, denominados "receptores", que definen un estado alternativo. Los estados ejecutan un receptor si encuentran errores en tiempo de ejecución y no tienen una política de reintentos o si su política de reintentos está agotada. Para obtener más información, consulte Estados alternativos.

Campos obsoletos

> **Note**
>
> Aunque puede seguir incluyendo estados `Map` que usen los siguientes campos, le recomendamos encarecidamente que sustituya `Iterator` por `ItemProcessor` y `Parameters` por `ItemSelector`.

`Iterator`

Especifica un objeto JSON que define un conjunto de pasos que procesan cada elemento de la matriz.

`Parameters`

Especifica una colección de pares clave-valor, donde los valores pueden contener lo siguientes:

- Valores estáticos se definen en la definición de la máquina de estado.
- Valores seleccionados de la entrada mediante una ruta.

Ejemplo de estado Map en línea

Tenga en cuenta los siguientes datos de entrada para un estado `Map` que se ejecute en modo En línea.

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

Dada la entrada anterior, el estado `Map` del siguiente ejemplo invocará una función AWS Lambda denominada `ship-val` una vez por cada elemento de la matriz en el campo `shipped`.

```
"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"
}
```

Cada iteración del estado `Map` enviará un elemento de la matriz. seleccionado mediante el campo `ItemsPath`, como entrada para la función de Lambda `ship-val`. Los siguientes valores son un ejemplo de la entrada que el estado `Map` envía a una invocación de la función de Lambda:

```
{
  "prod": "R31",
```

```
  "dest-code": 9511,
  "quantity": 1344
}
```

Una vez completada, la salida del estado `Map` es una matriz JSON en la que cada elemento es la salida de una iteración. En este caso, esta matriz contiene la salida de la función de Lambda `ship-val`.

Ejemplo de estado Map en línea con **`ItemSelector`**

Supongamos que la función de Lambda `ship-val` del ejemplo anterior también necesita información sobre el transportista del envío. Esta información se añade a los elementos de la matriz para cada iteración. Puede incluir información de la entrada, junto con información específica de la iteración actual del estado `Map`. Observe el campo `ItemSelector` del siguiente ejemplo:

```
"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
}
```

El bloque `ItemSelector` reemplaza la entrada a las iteraciones por un nodo JSON. Este nodo contiene tanto los datos del elemento actual del objeto de contexto como la información del transportista mensajería del campo `delivery-partner` de la entrada del estado `Map`. A

continuación, se muestra un ejemplo de entrada en una sola iteración. El estado `Map` pasa esta entrada a una invocación de la función de Lambda `ship-val`.

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

En el ejemplo anterior del estado Map en línea, el campo `ResultPath` produce una salida en el mismo formato que la entrada. No obstante, sobrescribe el campo `detail.shipped` con una matriz en la que cada elemento es la salida de la invocación de Lambda `ship-val` de cada iteración.

Para obtener más información sobre el uso del estado Map en línea y sus campos, consulte lo siguiente.

- Repetir una acción utilizando el estado Inline Map
- Procesamiento de entrada y salida en Step Functions
- ItemsPath
- Datos del objeto de contexto para los estados Map

Procesamiento de entrada y salida del estado **`Map`** en línea

Para un estado `Map` determinado, `InputPath` selecciona un subconjunto de la entrada del estado.

La entrada de un estado `Map` debe incluir una matriz JSON. El estado `Map` ejecuta la sección `ItemProcessor` una vez para cada elemento de la matriz. Si especifica el campo `ItemsPath`, el estado `Map` selecciona en qué parte de la entrada se busca la matriz sobre la que se va a iterar. Si no se especifica, el valor de `ItemsPath` es $ y la sección `ItemProcessor` espera que la matriz sea la única entrada. Si especifica el campo `ItemsPath`, su valor debe ser una ruta de referencia. El estado `Map` aplica esta ruta a la entrada efectiva después de aplicar la `InputPath`. La `ItemsPath` debe identificar un campo cuyo valor sea una matriz JSON.

La entrada de cada iteración, de forma predeterminada, es un único elemento del campo de matriz identificado por el valor `ItemsPath`. Puede anular este valor con el campo `ItemSelector`.

Una vez completada, la salida del estado `Map` es una matriz JSON en la que cada elemento es la salida de una iteración.

Para obtener más información acerca de las entradas y salidas del estado Map en línea, consulte lo siguiente:

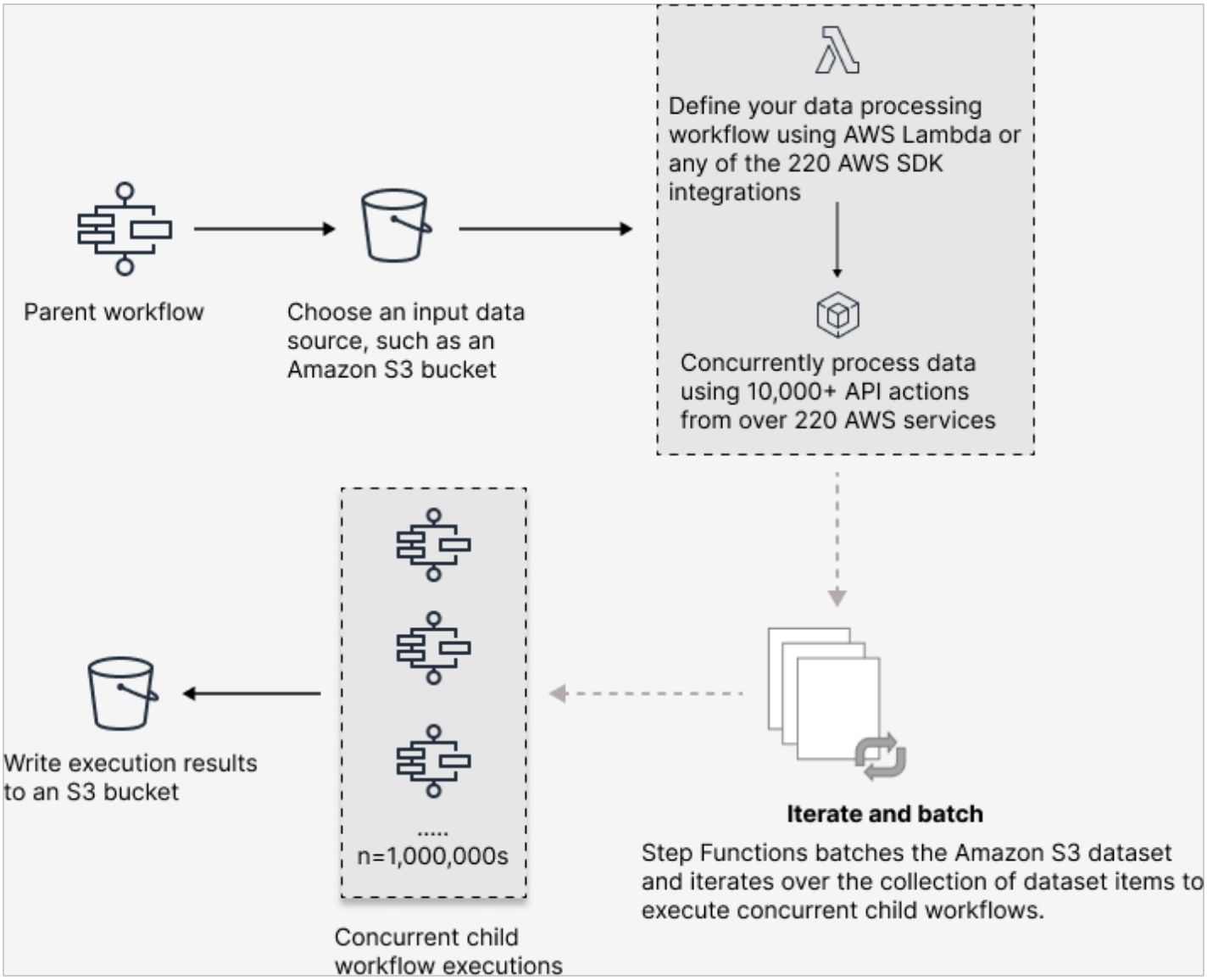
- Repetir una acción utilizando el estado Inline Map
- Ejemplo de estado Map en línea con `ItemSelector`
- Procesamiento de entrada y salida en Step Functions
- Datos del objeto de contexto para los estados Map
- Procesamiento dinámico de datos con un estado Map

Uso del estado Map en modo distribuido para orquestar cargas de trabajo paralelas a gran escala

Con Step Functions, puede orquestar cargas de trabajo paralelas a gran escala para realizar tareas, como el procesamiento bajo demanda de datos semiestructurados. Estas cargas de trabajo paralelas permiten procesar simultáneamente orígenes de datos a gran escala almacenados en Amazon S3. Por ejemplo, puede procesar un único archivo JSON o CSV que contenga grandes cantidades de datos. O bien, puede procesar un conjunto grande de objetos de Amazon S3.

Para configurar una carga de trabajo paralela a gran escala en los flujos de trabajo, incluya un estado `Map` en modo distribuido. El estado Map procesa los elementos de un conjunto de datos de forma simultánea. Un estado `Map` establecido en Distributed se conoce como estado Map Distributed. En el modo Distribuido, el estado `Map` permite un procesamiento de alta simultaneidad. En el modo Distribuido, el estado `Map` procesa los elementos del conjunto de datos en iteraciones llamadas ejecuciones de flujos de trabajo secundarios. Puede especificar el número de ejecuciones de flujo de trabajo secundario que se pueden ejecutar en paralelo. Cada ejecución de flujo de trabajo secundario tiene su propio historial de ejecución independiente del flujo de trabajo principal. Si no se especifica, Step Functions ejecuta 10 000 ejecuciones de flujos de trabajo secundarios en paralelo.

La siguiente ilustración explica cómo puede configurar cargas de trabajo paralelas a gran escala en los flujos de trabajo.



En este tema

Términos clave

Modo distribuido

Un modo de procesamiento del estado Map. En este modo, cada iteración del estado `Map` se ejecuta como una ejecución de flujo de trabajo secundario que permite una alta simultaneidad. Cada ejecución de flujo de trabajo secundario tiene su propio historial de ejecución, independiente del historial de ejecución del flujo de trabajo principal. Este modo admite la lectura de orígenes de datos de Amazon S3 a gran escala.

Estado Map Distributed

Un estado de Map establecido en modo de procesamiento distribuido.

Flujo de trabajo de Map

Un conjunto de pasos que ejecuta un estado `Map`.

Flujo de trabajo principal

Un flujo de trabajo que contiene uno o más estados Map Distributed.

Ejecución de flujo de trabajo secundario

Una iteración del estado Map Distributed. La ejecución de un flujo de trabajo secundario tiene su propio historial de ejecución, que es independiente del historial de ejecución del flujo de trabajo principal.

Map Run

Cuando se ejecuta un estado `Map` en modo distribuido, Step Functions crea un recurso Map Run. Map Run hace referencia a un conjunto de ejecuciones de flujos de trabajo secundarios que inicia un estado Map Distributed y a la configuración de tiempo de ejecución que controla estas ejecuciones. Step Functions asigna un nombre de recurso de Amazon (ARN) al Map Run. Puede examinar un Map Run en la consola de Step Functions. También puede invocar la acción de la API `DescribeMapRun`. A Map Run también emite métricas a CloudWatch.

Para obtener más información, consulte Examen de Map Run.

Ejemplo de definición de estado Map Distributed

Utilice el estado `Map` en modo Distribuido cuando necesite orquestar cargas de trabajo paralelas a gran escala que cumplan cualquier combinación de las siguientes condiciones:

- El tamaño del conjunto de datos supera los 256 KB.
- El historial de eventos de ejecución del flujo de trabajo supera las 25 000 entradas.
- Necesita una simultaneidad de más de 40 iteraciones paralelas.

En el siguiente ejemplo de definición de estado Map Distributed, se especifica el conjunto de datos como un archivo CSV almacenado en un bucket de Amazon S3. También especifica una función de Lambda que procesa los datos de cada fila del archivo CSV. Como en este ejemplo se utiliza un archivo CSV, también se especifica la ubicación de los encabezados de las columnas CSV. Para ver la definición completa de la máquina de estado de este ejemplo, consulte el tutorial Copia de datos CSV a gran escala mediante Map Distributed.

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

Permisos para ejecutar Map Distributed

Cuando se incluye un estado Map Distributed en los flujos de trabajo, Step Functions necesita los permisos adecuados para permitir que el rol de la máquina de estado invoque la acción de la API `StartExecution` para el estado Map Distributed.

El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios al rol de la máquina de estado para ejecutar el estado Map Distributed.

> **Note**
>
> No olvide reemplazar *stateMachineName* por el nombre de la máquina de estado en la que está utilizando el estado Map Distributed. Por ejemplo, `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:*"
    }
  ]
}
```

Además, debe asegurarse de tener los privilegios mínimos necesarios para acceder a los AWS recursos utilizados en el estado del mapa distribuido, como los buckets de Amazon S3. Para obtener más información, consulte Políticas de IAM para usar el estado Map Distributed.

Campos de estado de Map Distributed

Para usar el estado Map Distributed en los flujos de trabajo, especifique uno o más de estos campos. Estos campos se especifican además de los campos de estado comunes.

`Type` (Obligatorio)

Establece el tipo de estado, por ejemplo `Map`.

`ItemProcessor` (Obligatorio)

Contiene los siguientes objetos JSON que especifican el modo y la definición de procesamiento del estado `Map`.

- `ProcessorConfig`: un objeto JSON que especifica la configuración del estado `Map`. Este objeto contiene los siguientes subcampos:
  - `Mode`: configurado en **`DISTRIBUTED`** para usar el estado `Map` en modo distribuido.

    > **Note**
    >
    > Actualmente, si se usa el estado `Map` dentro de flujos de trabajo rápidos, no se puede configurar el `Mode` en `DISTRIBUTED`. Sin embargo, si se usa el estado `Map` dentro de flujos de trabajo estándar, se puede configurar el `Mode` en `DISTRIBUTED`.

- `ExecutionType` – Especifica el tipo de ejecución del flujo de trabajo de Map como STANDARD o EXPRESS. Debe proporcionar este campo si especificó `DISTRIBUTED` para el subcampo `Mode`. Para obtener más información acerca de los tipos de flujo de trabajo, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

- `StartAt`: especifica una cadena que indica el primer estado de un flujo de trabajo. Esta cadena debe coincidir exactamente (mayúsculas y minúsculas) con el nombre de uno de los objetos de estado. Este estado se ejecuta primero para cada elemento del conjunto de datos. Cualquier entrada de ejecución que se proporcione al estado `Map` pasará primero al estado `StartAt`.

- `States` – Objeto JSON que contiene un conjunto de estados delimitados por comas. En este objeto, se define el Map workflow.

`ItemReader`

Especifica un conjunto de datos y su ubicación. El estado `Map` recibe sus datos de entrada del conjunto de datos especificado.

En el modo distribuido, puede utilizar como conjunto de datos una carga JSON transferida desde un estado anterior o un origen de datos de Amazon S3 a gran escala. Para obtener más información, consulte ItemReader.

`ItemsPath` (Opcional)

Especifica una ruta de referencia mediante la JsonPathsintaxis para seleccionar el nodo JSON que contiene una matriz de elementos dentro de la entrada de estado.

En el modo Distribuido, solo se especifica este campo cuando se utiliza una matriz JSON de un paso anterior como entrada de estado. Para obtener más información, consulte ItemsPath.

`ItemSelector` (Opcional)

Anula los valores de los elementos individuales del conjunto de datos antes de pasarlos a cada iteración del estado `Map`.

En este campo, se especifica una entrada JSON válida que contiene un conjunto de pares clave-valor. Estos pares pueden ser valores estáticos que se establezcan en la definición de la máquina de estado, valores seleccionados de la entrada de estado mediante una ruta o valores a los que se accede desde el objeto de contexto. Para obtener más información, consulte ItemSelector.

`ItemBatcher` (Opcional)

Especifica el procesamiento de los elementos del conjunto de datos por lotes. Cada ejecución de flujo de trabajo secundario recibe entonces un lote de estos elementos como entrada. Para obtener más información, consulte ItemBatcher.

`MaxConcurrency` (Opcional)

Especifica el número de ejecuciones de flujo de trabajo secundario que se pueden ejecutar en paralelo. El intérprete solo permite el número especificado de ejecuciones de flujos de trabajo secundarios paralelos. Si no se especifica un valor de simultaneidad o se establece en cero, Step Functions no limita la simultaneidad y realiza 10 000 ejecuciones paralelas de flujos de trabajo secundarios.

> **Note**
>
> Si bien puede especificar un límite de simultaneidad superior para las ejecuciones de flujos de trabajo secundarios en paralelo, le recomendamos que no exceda la capacidad de un AWS servicio descendente, como. AWS Lambda

`MaxConcurrencyPath` (Opcional)

Si desea proporcionar un valor máximo de simultaneidad de forma dinámica a partir de la entrada de estado mediante una ruta de referencia, utilice `MaxConcurrencyPath`. Una vez resuelta, la ruta de referencia debe seleccionar un campo cuyo valor sea un entero no negativo.

> **Note**
>
> Un estado `Map` no puede incluir tanto `MaxConcurrency` como `MaxConcurrencyPath`.

`ToleratedFailurePercentage` (Opcional)

Define el porcentaje de elementos con error que se toleran en un Map Run. El Map Run genera automáticamente un error si se supera este porcentaje. Step Functions calcula el porcentaje de elementos con error como resultado del número total de elementos con error o con tiempo de espera agotado dividido por el número total de elementos. Debe especificar un valor comprendido entre cero y 100. Para obtener más información, consulte Umbral de error tolerado para el estado Distributed Map.

`ToleratedFailurePercentagePath` (Opcional)

Si desea proporcionar un valor de porcentaje de error tolerado de forma dinámica a partir de la entrada de estado con una ruta de referencia, utilice `ToleratedFailurePercentagePath`. Una vez resuelta, la ruta de referencia debe seleccionar un campo cuyo valor esté entre cero y 100.

`ToleratedFailureCount` (Opcional)

Define el número de elementos con error que se toleran en un Map Run. El Map Run genera automáticamente un error si se supera este porcentaje. Para obtener más información, consulte Umbral de error tolerado para el estado Distributed Map.

`ToleratedFailureCountPath` (Opcional)

Si desea proporcionar un valor de recuento de error tolerado de forma dinámica a partir de la entrada de estado con una ruta de referencia, utilice `ToleratedFailureCountPath`. Una vez resuelta, la ruta de referencia debe seleccionar un campo cuyo valor sea un entero no negativo.

`Label` (Opcional)

Una cadena que identifica de forma exclusiva un estado `Map`. Para cada Map Run, Step Functions añade la etiqueta al ARN de Map Run. A continuación se muestra un ejemplo de un ARN de Map Run con una etiqueta personalizada denominada `demoLabel`:

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

Si no especifica una etiqueta, Step Functions genera automáticamente una etiqueta única.

> **Note**
>
> Las etiquetas no pueden tener más de 40 caracteres de longitud, deben ser únicas dentro de una definición de máquina de estado y no pueden contener ninguno de los siguientes caracteres:
>
> - Caracteres de espacio en blanco
> - Caracteres comodín (`? *`)
> - Caracteres entre corchetes (`< > { } [ ]`)
> - Caracteres especiales (`: ; , \ | ^ ~ $ # % & ` "`)
> - Caracteres de control (`\\u0000` - `\\u001f` o `\\u007f` - `\\u009f`).

Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

`ResultWriter` (Opcional)

Especifica la ubicación de Amazon S3 en la que Step Functions escribe todos los resultados de ejecución de flujo de trabajo secundario.

Step Functions consolida todos los datos de ejecución del flujo de trabajo secundario, como la entrada y salida de la ejecución, el ARN y el estado de ejecución. A continuación, exporta las ejecuciones con el mismo estado a sus archivos respectivos en la ubicación de Amazon S3 especificada. Para obtener más información, consulte ResultWriter.

Si no se exportan los resultados del estado `Map`, devolverá una matriz con todos los resultados de la ejecución del flujo de trabajo secundario. Por ejemplo:

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

`ResultPath` (Opcional)

Especifica en qué lugar de la entrada se va a situar la salida de las iteraciones. La entrada se filtra entonces según el contenido del campo `OutputPath`, si está presente, antes de que se pase como salida del estado. Para obtener más información, consulte Procesamiento de entrada y salida.

`ResultSelector` (Opcional)

Pase un conjunto de pares clave-valor, donde los valores sean estáticos o se seleccionen del resultado. Para obtener más información, consulte ResultSelector.

> **Tip**
>
> Si el estado Parallel o Map que utiliza en sus máquinas de estado devuelve una matriz de matrices, puede transformarlas en una matriz plana con el campo ResultSelector. Para obtener más información, consulte Aplanamiento de una matriz de matrices.

`Retry` (Opcional)

Una matriz de objetos, denominados "reintentadores", que definen una política de reintentos. Una ejecución utiliza la política de reintento si el estado encuentra errores en tiempo de ejecución. Para obtener más información, consulte Ejemplos de máquina de estado que usan Retry y Catch.

> **Note**
>
> Si define "reintentadores" para el estado Map Distributed, la política de reintentos se aplica a todas las ejecuciones de flujo de trabajo secundario que haya iniciado el estado `Map`. Por ejemplo, imagine que el estado `Map` ha iniciado tres ejecuciones de flujos de trabajo secundarios, de las cuales una genera un error. Cuando se produce el error, la ejecución utiliza el campo `Retry`, si está definido, para el estado `Map`. La política de reintentos se aplica a todas las ejecuciones del flujo de trabajo secundario y no solo a las ejecuciones con error. Si se produce un error en la ejecución de uno o más flujos de trabajo secundarios, se produce un error en el Map Run.
> Al reintentar un estado `Map`, se crea un nuevo Map Run.

`Catch` (Opcional)

Una matriz de objetos, denominados "receptores", que definen un estado alternativo. Step Functions usa los "receptores" definidos en `Catch` si el estado encuentra errores en tiempo de ejecución. Cuando se produce un error, la ejecución utiliza primero los "reintentadores" definidos en `Retry`. Si la política de reintentos no está definida o está agotada, la ejecución utiliza sus "receptores", si están definidos. Para obtener más información, consulte Estados alternativos.

Siguientes pasos

Para obtener más información acerca del estado Map Distributed, consulte los siguientes recursos:

- Procesamiento de entrada y salida

  Para configurar la entrada que recibe un estado Map Distributed y la salida que genera, Step Functions proporciona los siguientes campos:

  - ItemReader
  - ItemsPath
  - ItemSelector

- ItemBatcher
- ResultWriter
- Análisis de un archivo CSV de entrada

  Además de estos campos, Step Functions también permite definir un umbral de error tolerado para Map Distributed. Este valor permite especificar el número máximo o el porcentaje de elementos con error como umbral de error para Map Run. Para obtener más información acerca de la configuración del umbral de error tolerado, consulte Umbral de error tolerado para el estado Distributed Map.

- Uso del estado Map Distributed

  Consulte los siguientes tutoriales y proyectos de muestra para empezar a utilizar el estado Map Distributed.

  - Introducción al uso del estado Distributed Map
  - Procesamiento de lotes completos de datos con una función de Lambda
  - Procesamiento de elementos de datos individuales con una función de Lambda
  - Proyecto de ejemplo: procesar un archivo CSV con Map Distributed
  - Proyecto de ejemplo: procesar datos de un bucket de Amazon S3 con Map Distributed

- Examen de la ejecución del estado Map Distributed

  La consola de Step Functions proporciona una página Detalles de Map Run que muestra toda la información relacionada con la ejecución de un estado Map Distributed. Para obtener información sobre cómo examinar la información que se muestra en esta página, consulte Examen de Map Run.

Umbral de error tolerado para el estado Distributed Map

Al orquestar cargas de trabajo paralelas a gran escala, también puede definir un umbral de error tolerado. Este valor permite especificar el número máximo o el porcentaje de elementos con error como umbral de error para Map Run. Según el valor que especifique, su Map Run generará un error automáticamente si supera el umbral. Si especifica ambos valores, el flujo de trabajo genera un error cuando supera alguno de los valores.

La especificación de un umbral contribuye a que se produzca un error en un número específico de elementos antes de que lo haga en toda la Map Run. Step Functions devuelve un error

`States.ExceedToleratedFailureThreshold` cuando la Map Run genera un error porque se ha superado el umbral especificado.

> **Note**
>
> Step Functions puede seguir ejecutando flujos de trabajo secundarios en una Map Run incluso después de superar el umbral de error tolerado, pero antes de que la Map Run genere un error.

Para especificar el valor de umbral en Workflow Studio, seleccione Definir un umbral de error tolerado en Configuración adicional en el campo Configuración del tiempo de ejecución.

Porcentaje de errores tolerados

Define el porcentaje de elementos con error que se van a tolerar. Su Map Run genera un error si se supera este valor. Step Functions calcula el porcentaje de elementos con error como resultado del número total de elementos con error o con tiempo de espera agotado dividido por el número total de elementos. Debe especificar un valor comprendido entre cero y 100. El valor porcentual predeterminado es cero, lo que significa que el flujo de trabajo genera un error si alguna de las ejecuciones de flujos de trabajo secundarios produce un error o se agota el tiempo de espera. Si especifica el porcentaje en 100, el flujo de trabajo no dará error aunque se produzcan errores en todas las ejecuciones de los flujos de trabajos secundarios.

Además, puede especificar el porcentaje como ruta de referencia a un par clave-valor existente en la entrada del estado Distributed Map. Esta ruta debe convertirse en un entero positivo entre 0 y 100 en tiempo de ejecución. La ruta de referencia se especifica en el subcampo `ToleratedFailurePercentagePath`.

Por ejemplo, en el caso de la entrada siguiente:

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

Puede especificar el porcentaje mediante una ruta de referencia a esa entrada de la siguiente manera:

```
{
```

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

> **Important**
>
> Puede especificar `ToleratedFailurePercentage` o `ToleratedFailurePercentagePath`, pero no los dos, en la definición del estado Map Distributed.

Recuento de errores tolerados

Define el número de elementos con error que se van a tolerar. Su Map Run genera un error si se supera este valor.

Además, puede especificar el recuento como ruta de referencia a un par clave-valor existente en la entrada del estado Distributed Map. Esta ruta debe convertirse en un número entero positivo en tiempo de ejecución. La ruta de referencia se especifica en el subcampo `ToleratedFailureCountPath`.

Por ejemplo, en el caso de la entrada siguiente:

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

Puede especificar el número mediante una ruta de referencia a esa entrada de la siguiente manera:

```
{
  ...
  "Map": {
    "Type": "Map",
    ...
```

```
    "ToleratedFailureCountPath": "$.count"
    ...
  }
}
```

> **Important**
>
> Puede especificar `ToleratedFailureCount` o `ToleratedFailureCountPath`, pero no los dos, en la definición del estado Distributed Map.

Transiciones

Cuando inicia una nueva ejecución de su máquina de estado, el sistema comienza con el estado al que se hace referencia en el campo principal `StartAt`. Este campo, que se proporciona como una cadena, debe coincidir exactamente, incluyendo mayúsculas y minúsculas, con el nombre de un estado en el flujo de trabajo.

Después de que se ejecute un estado, AWS Step Functions utiliza el valor del campo `Next` para determinar el siguiente estado al que debe avanzar.

Los campos `Next` también especifican los nombres de estado como cadenas. Esta cadena distingue entre mayúsculas y minúsculas y debe coincidir exactamente con el nombre de un estado definido en la descripción de la máquina de estado

Por ejemplo, el siguiente estado incluye una transición a `NextState`.

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

La mayoría de los estados solamente permiten usar una regla de transición con el campo `Next`. Sin embargo, algunos estados de control de flujo, como el estado `Choice`, permiten especificar varias reglas de transición, cada una con su propio campo `Next`. El lenguaje de estados de Amazon proporciona detalles sobre cada uno de los tipos de estado que se pueden especificar; por ejemplo, información acerca de cómo especificar las transiciones.

Los estados pueden tener varias transiciones de entrada procedentes de otros estados.

El proceso se repite hasta que alcanza un estado terminal (un estado con `"Type": Succeed`, `"Type": Fail` o `"End": true`) o se produce un error del sistema en tiempo de ejecución.

Cuando redrive una ejecución, se considera una transición de estado. Además, todos los estados que se vuelven a ejecutar en un redrive también se consideran transiciones de estado.

Las siguientes reglas se aplican a los estados de una máquina de estado:

- Los estados pueden tener lugar en cualquier orden dentro del bloque delimitado. No obstante, el orden en el que aparecen no afecta al orden en el que se ejecuta. El contenido de los estados determina este orden.
- En una máquina de estado solo puede haber un estado designado como estado `start`. El estado `start` se define mediante el valor del campo `StartAt` en la estructura de nivel superior.
- En función de la lógica de la máquina de estado (por ejemplo, si la máquina de estado tiene varias ramificaciones lógicas), es posible que haya varios estados `end`.
- Si la máquina de estado se compone de un solo estado, este podría ser tanto el estado inicial como el estado final.

Transiciones en el estado Distributed Map

Cuando utilice el estado `Map` en modo Distributed, se le cobrará una transición de estado por cada ejecución del flujo de trabajo secundario que inicie el estado Distributed Map. Al usar el estado `Map` en el modo En línea, no se carga una transición de estado por cada iteración del estado Map en línea.

Puede optimizar los costes mediante el estado `Map` en modo distribuido e incluir un flujo de trabajo anidado en la definición del estado `Map`. El estado Distributed Map también agrega más valor al iniciar ejecuciones de flujos de trabajo secundarios del tipo Rápido. Step Functions almacena la respuesta y el estado de las ejecuciones de flujos de trabajo secundarios rápidos, lo que reduce la necesidad de almacenar los datos de ejecución en CloudWatch Logs. También puede acceder a los controles de flujo disponibles con un estado Distributed Map, como definir umbrales de error o agrupar por lotes un grupo de elementos. Para obtener más información acerca de los precios de Step Functions, consulte Precios de AWS Step Functions.

Datos de la máquina de estado

Los datos de la máquina de estado son los siguientes:

- Los datos de entrada iniciales de la máquina de estado
- Los datos que se pasan entre estados
- La salida de la máquina de estado

En esta sección, se describe el formato de los datos de la máquina de estado y cómo se utilizan estos datos en AWS Step Functions.

Temas

Formato de los datos

Los datos de la máquina de estado se representan mediante texto JSON. Puede proporcionar valores a una máquina de estado con cualquier tipo de datos compatible con JSON.

> **Note**
>
> - Los números en formato de texto JSON se ajustan a la semántica de JavaScript. Por lo general, estos números se corresponden con valores IEEE-854 de doble precisión.
> - A continuación se muestra texto JSON válido:
>   - Cadenas independientes, delimitadas por comillas
>   - Objetos
>   - Matrices
>   - Números
>   - Valores booleanos
>   - `null`
> - La salida de un estado se convierte en la entrada para el siguiente estado. Sin embargo, con el procesamiento de entrada y salida, puede restringir los estados para que trabajen con un subconjunto de los datos de entrada.

Entrada y salida de la máquina de estado

Puede proporcionar los datos de entrada iniciales a una máquina de estado AWS Step Functions de dos maneras. Puede pasar los datos a una acción `StartExecution` al iniciar una ejecución. También puede pasar los datos a la máquina de estado desde la consola de Step Functions. Los datos iniciales se pasan al estado `StartAt` de la máquina de estado. Si no se proporcionan datos de entrada, el valor predeterminado es un objeto vacío (`{}`).

El último estado (`terminal`) es el encargado de devolver la salida de la ejecución. Esta salida aparece como un texto JSON en el resultado de la ejecución.

Para los flujos de trabajo estándar, puede recuperar los resultados del historial de ejecuciones mediante intermediarios externos, tales como la acción `DescribeExecution`. Puede consultar los resultados de la ejecución en la consola de Step Functions.

En el caso de los flujos de trabajo rápidos, si ha activado el registro puede recuperar los resultados de CloudWatch Logs o ver y depurar las ejecuciones en la consola de Step Functions. Para obtener más información, consulte Registro mediante CloudWatch Logs y Visualización y depuración de ejecuciones en la consola de Step Functions.

También debe tener en cuenta las cuotas relacionadas con la máquina de estado. Para obtener más información, consulte Cuotas

Entrada y salida de estados

Cada entrada de estado se compone de un texto JSON procedente del estado anterior o del estado `StartAt`, la entrada que activa la ejecución. Algunos estados de control de flujo replican los datos de entrada en la salida.

En el ejemplo siguiente, la máquina de estado suma dos números.

1. Defina la función de AWS Lambda.

```
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. Defina la máquina de estado de .

```
{
  "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. Inicie una ejecución con el siguiente texto JSON.

```
{ "numbers": [3, 4] }
```

El estado Add recibe el texto JSON y lo pasa a la función de Lambda.

La función de Lambda devuelve el resultado del cálculo al estado.

El estado devuelve el siguiente valor en la salida.

```
{ "result": 7 }
```

Como Add también es el estado final de la máquina de estado, este valor se devuelve como salida de la máquina de estado.

Si el estado final no devuelve ninguna salida, la máquina de estado devuelve un objeto vacío ({}).

Para obtener más información, consulte Procesamiento de entrada y salida en Step Functions.

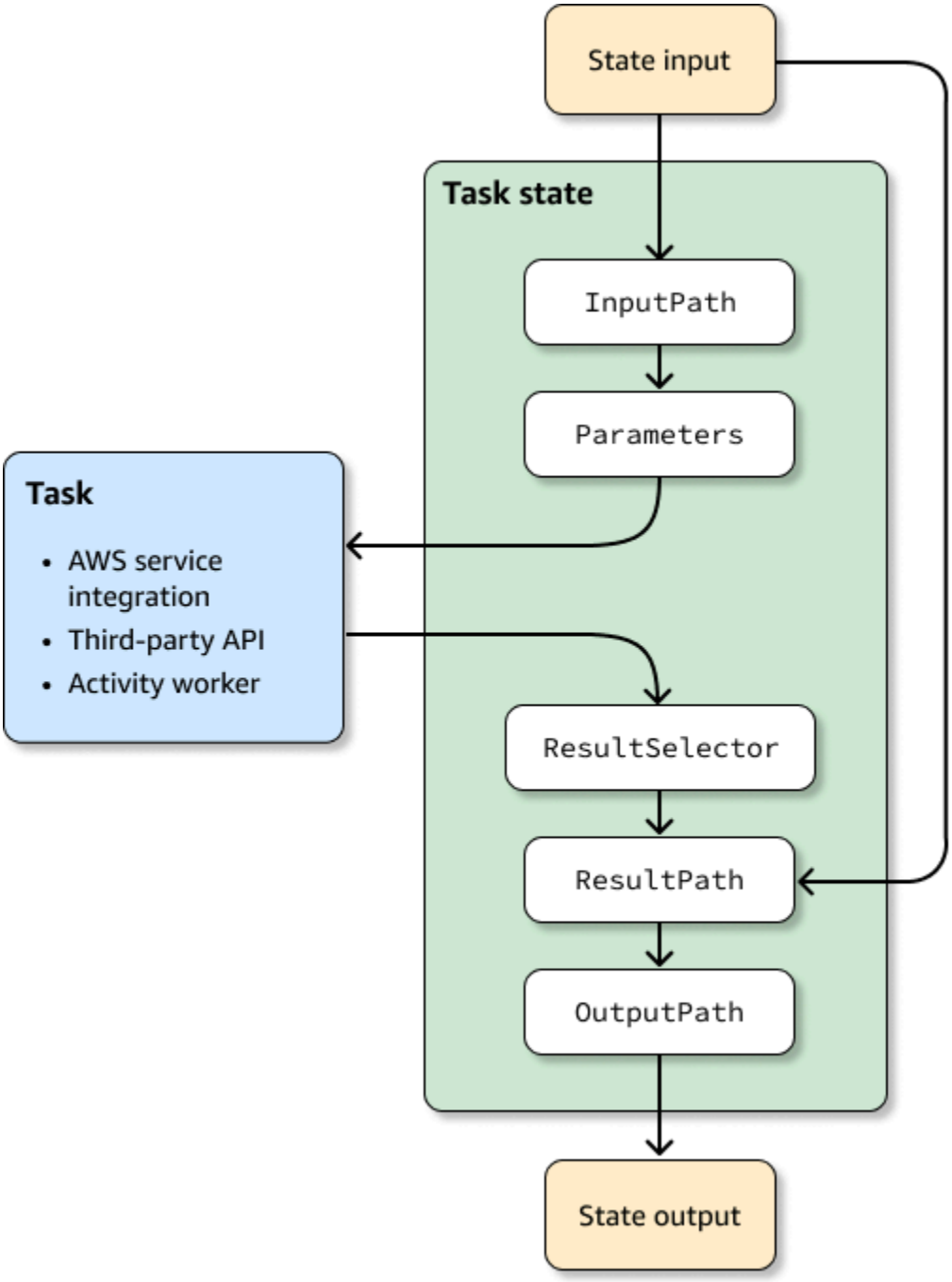
Procesamiento de entrada y salida en Step Functions

Una ejecución de Step Functions recibe un texto JSON como entrada y transfiere dicha entrada al primer estado en el flujo de trabajo. Los estados individuales reciben JSON como entrada y normalmente pasan JSON como salida al siguiente estado. Comprender cómo fluye esta información de estado a estado y aprender a filtrar y manipular estos datos resulta esencial para diseñar e implementar de forma eficaz los flujos de trabajo en AWS Step Functions.

En Amazon States Language, estos campos filtran y controlan el flujo de JSON de estado a estado:

- `InputPath`
- `Parameters`
- `ResultSelector`
- `ResultPath`
- `OutputPath`

El siguiente diagrama muestra cómo se mueve la información de JSON a través del estado de una tarea. `InputPath`selecciona qué partes de la entrada JSON pasarán a la tarea del `Task` estado (por ejemplo, una AWS Lambda función). `ResultPath`a continuación, selecciona qué combinación de la entrada de estado y el resultado de la tarea desea pasar a la salida. `OutputPath`puede filtrar la salida JSON para limitar aún más la información que se pasa a la salida.



`InputPath`, `Parameters`, `ResultSelector`, `ResultPath` y `OutputPath` manipulan el objeto JSON a medida que se desplaza a través de cada estado del flujo de trabajo.

Cada uno de ellos puede utilizar rutas para seleccionar partes del objeto JSON de la entrada o del resultado. Una ruta es una cadena, que empieza por $ e identifica nodos dentro de texto JSON. Las rutas de Step Functions utilizan JsonPathla sintaxis.

> **Tip**
>
> Utilice el simulador de flujo de datos de la consola de Step Functions para probar la sintaxis de las rutas JSON, comprender mejor cómo se manipulan los datos dentro de un estado y ver cómo se transfieren los datos entre los estados.

> **Tip**
>
> Para implementar un ejemplo de un flujo de trabajo que incluye el procesamiento de entradas y salidas Cuenta de AWS, consulte el Módulo 6: Procesamiento de entradas y salidas de The AWS Step Functions Workshop.

Temas

Rutas

En Amazon States Language, una ruta es una cadena que comienza por $ y se puede utilizar para identificar componentes en el texto JSON. Las rutas siguen JsonPathla sintaxis. Puede especificar una ruta para acceder a los subconjuntos de la entrada al especificar valores para `InputPath`, `ResultPath` y `OutputPath`. Para más información, consulte Procesamiento de entrada y salida en Step Functions.

> **Note**
>
> También puede especificar un nodo de JSON de la entrada o del objeto context mediante rutas en el campo `Parameters` de una definición de estado. Consulte Cómo pasar parámetros a una API de servicio.

Debe utilizar la notación entre corchetes si el nombre de campo contiene algún carácter que no esté incluido en la `member-name-shorthand` definición de la regla JsonPath ABNF. Por lo tanto, para codificar caracteres especiales, como los signos de puntuación (excepto_), debe utilizar la notación entre corchetes. Por ejemplo, `$.abc.['def ghi']`.

Rutas de referencia

Una ruta de referencia es una ruta cuya sintaxis está limitada de tal forma que solo puede identificar un único nodo en una estructura JSON:

- Solo puede obtener acceso a los campos de objeto con la notación de punto (`.`) y corchete (`[ ]`).
- Funciones como `length()` no son compatibles.
- Los operadores léxicos, que no son simbólicos, como `subsetof` no están admitidos.
- No se admite el filtrado por expresiones regulares o por referencia a otro valor de la estructura JSON.
- Los operadores`@`, `,`, `:`, y `?` no son compatibles

Por ejemplo, si los datos de entrada del estado contienen los valores siguientes:

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  }
}
```

Las siguientes rutas de referencia devolverían lo siguiente.

```
$.foo => 123
$.bar => ["a", "b", "c"]
```

```
$.car.cdr => true
```

Algunos estados utilizan rutas y rutas de referencia para controlar el flujo de una máquina de estado o para configurar los valores o las opciones del estado. Para obtener más información, consulte Modelar el procesamiento de las rutas de entrada y salida del flujo de trabajo con un simulador de flujo de datos y Utilizar JSONPath de forma eficaz en. AWS Step Functions

Aplanamiento de una matriz de matrices

Si el estado Parallel o Map de las máquinas de estado devuelve una matriz de matrices, puede transformarlas en una matriz plana con el campo ResultSelector. Puede incluir este campo dentro de la definición de estados Parallel o Map para manipular el resultado de estos estados.

Para aplanar matrices, utilice la sintaxis JMESPath `[*]` en el campo `ResultSelector`, como se muestra en el siguiente ejemplo.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Para ver ejemplos que muestran cómo aplanar una matriz, consulte el Paso 3 de los siguientes tutoriales:

- Procesamiento de lotes completos de datos con una función de Lambda
- Procesamiento de elementos de datos individuales con una función de Lambda

InputPath, Parámetros y ResultSelector

Los campos `InputPath`, `Parameters` y `ResultSelector` proporcionan una forma de manipular JSON a medida que se mueve a través del flujo de trabajo. `InputPath` puede limitar la entrada que se mueve mediante el filtrado de la notación JSON a través de una ruta (consulte Rutas). El campo `Parameters` permite superar un conjunto de pares de clave-valor, donde los valores son valores estáticos que se definen en la definición de máquina de estado o que se seleccionan de la entrada mediante una ruta. El campo `ResultSelector` proporciona una forma de manipular el resultado del estado antes de aplicar `ResultPath`.

AWS Step Functions aplica primero el `InputPath` campo y, después, el `Parameters` campo. Puede filtrar primero la entrada sin procesar para obtener la selección que desee mediante `InputPath` y, a continuación, aplicar `Parameters` para continuar manipulando esa entrada o para

añadir valores nuevos. A continuación, puede utilizar el campo `ResultSelector` para manipular la salida del estado antes de aplicar `ResultPath`.

> **Tip**
>
> Utilice el simulador de flujo de datos de la consola de Step Functions para probar la sintaxis de las rutas JSON, comprender mejor cómo se manipulan los datos dentro de un estado y ver cómo se transfieren los datos entre los estados.

InputPath

Utilice `InputPath` para seleccionar una parte de la entrada del estado.

Por ejemplo, suponga que la entrada del estado incluye lo siguiente:

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Puede aplicar esta `InputPath`.

```
"InputPath": "$.dataset2",
```

Con dicha `InputPath`, lo siguiente es el objeto JSON que se pasa como la entrada.

```
{
  "val1": "a",
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> Una ruta puede generar una selección de valores. Considere el siguiente ejemplo.
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> Si aplica la ruta `$.a[0:2]`, el resultado es el siguiente.
>
> ```
> [ 1, 2 ]
> ```

Parámetros

En esta sección se describen las diferentes formas en las que puede utilizar el campo Parameters.

Pares clave-valor

Utilice el campo `Parameters` para crear una colección de pares clave-valor que se pasan como entrada. Los valores pueden ser valores estáticos que se incluyen en la definición de la máquina de estado, o pueden seleccionarse en la entrada o en el objeto de contexto con una ruta. Para los pares clave-valor cuyo valor se selecciona mediante una ruta, el nombre de la clave debe terminar por `.$`.

Por ejemplo, suponga que proporciona la siguiente entrada.

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Para seleccionar parte de la información, puede especificar estos parámetros en la definición de la máquina de estado.

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

Con la entrada anterior y el campo `Parameters`, este es el objeto JSON que se pasa.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

Además de la adición a la entrada, puede acceder a un objeto JSON especial conocido como el objeto de contexto. El objeto de contexto incluye información acerca de la ejecución de la máquina de estado. Consulte Objeto Context (Contexto).

Recursos conectados

El campo `Parameters` también puede pasar información a recursos conectados. Por ejemplo, si el estado de tu tarea es organizar un AWS Batch trabajo, puedes pasar los parámetros de la API pertinentes directamente a las acciones de la API de ese servicio. Para obtener más información, consulte:

- Cómo pasar parámetros a una API de servicio
- Trabajo con otros servicios

Amazon S3

Si los datos de la función de Lambda que pasa de un estado a otro pueden crecer hasta superar los 262 144 bytes, le recomendamos que utilice Amazon S3 para almacenar los datos e implementar uno de los siguientes métodos:

- Utilice el estado Map Distributed en el flujo de trabajo para que el estado `Map` pueda leer la entrada directamente de las fuentes de datos de Amazon S3. Para obtener más información, consulte Uso del estado Map en modo distribuido.
- Analice el nombre de recurso de Amazon (ARN) del bucket en el parámetro `Payload` para obtener el nombre de bucket y el valor de clave. Para obtener más información, consulte Utilizar los ARN de Amazon S3 en lugar de pasar cargas de gran tamaño.

También puede ajustar la implementación para que se pasen cargas más pequeñas en las ejecuciones.

ResultSelector

Utilice el campo `ResultSelector` para manipular el resultado de un estado antes de aplicar `ResultPath`. El campo `ResultSelector` permite crear una colección de pares de clave-valor, donde los valores son estáticos o se seleccionan del resultado del estado. Con el campo `ResultSelector`, puede elegir qué partes del resultado de un estado desea pasar al campo `ResultPath`.

> **Note**
>
> Con el campo `ResultPath`, puede añadir la salida del campo `ResultSelector` a la entrada original.

`ResultSelector` es un campo opcional en los siguientes estados:

- Map
- Estado de la tarea
- Parallel

Por ejemplo, las integraciones de servicios de Step Functions devuelven metadatos además de la carga del resultado. `ResultSelector` puede seleccionar partes del resultado y combinarlas con la entrada de estado con `ResultPath`. En este ejemplo, queremos seleccionar solo `resourceType` y `ClusterId` y combinar eso con la entrada de estado de un createCluster.sync de Amazon EMR. Dado lo siguiente:

```
{
```

```
  "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"
  }
}
```

Puede seleccionar entonces `resourceType` y `ClusterId` mediante `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"
}
```

Con la entrada dada, el uso de `ResultSelector` produce:

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
```

```
}
```

Aplanamiento de una matriz de matrices

Si el estado Parallel o Map de las máquinas de estado devuelve una matriz de matrices, puede transformarlas en una matriz plana con el campo ResultSelector. Puede incluir este campo dentro de la definición de estados Parallel o Map para manipular el resultado de estos estados.

Para aplanar matrices, utilice la sintaxis JMESPath `[*]` en el campo `ResultSelector`, como se muestra en el siguiente ejemplo.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Para ver ejemplos que muestran cómo aplanar una matriz, consulte el Paso 3 de los siguientes tutoriales:

- Procesamiento de lotes completos de datos con una función de Lambda
- Procesamiento de elementos de datos individuales con una función de Lambda

ResultPath

La salida de un estado puede ser una copia de su entrada, el resultado que produce (por ejemplo, la salida de una función de Lambda del estado `Task`) o una combinación de su entrada y del resultado. Use `ResultPath` para controlar qué combinación de estos se pasa a la salida del estado.

Los siguientes tipos de estado pueden generar un resultado y pueden incluir `ResultPath`:

- Pass
- Estado de la tarea
- Parallel
- Map

Use `ResultPath` para combinar un resultado de tarea con una entrada de tarea o para seleccionar uno de estos. La ruta que se proporciona a `ResultPath` controla la información que se pasa a la salida.

> **Note**
>
> `ResultPath` se limita al uso de rutas de referencia, que limitan el ámbito de forma que solo pueden identificar un único nodo en JSON. Consulte Rutas de referencia en Amazon States Language.

Estos ejemplos se basan en la máquina de estado y en la función de Lambda que se describen en el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda. Realice el tutorial y ensaye las distintas salidas probando diversas rutas en un campo `ResultPath`.

Use ResultPath para:

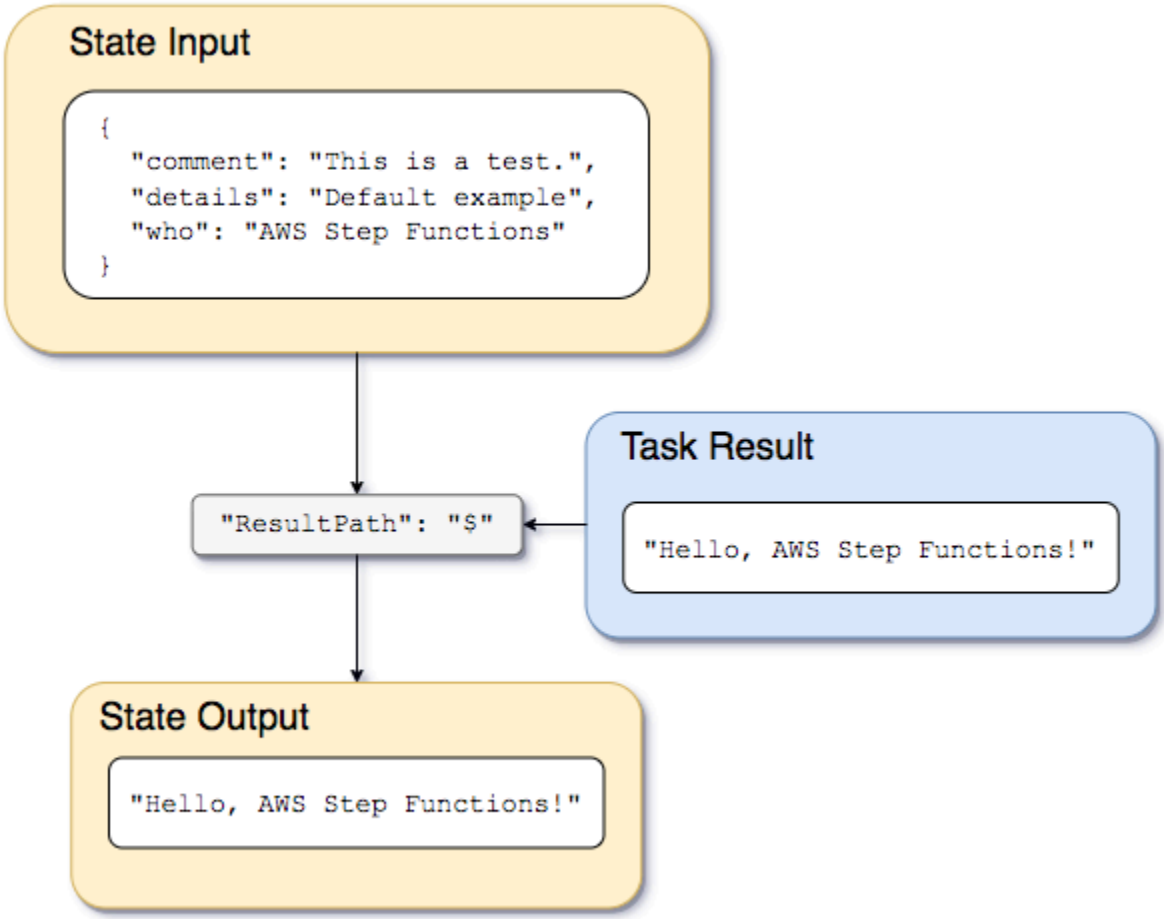
- Se usa ResultPath para reemplazar la entrada con el resultado
- Descartar el resultado y conservar la entrada original
- Se utiliza ResultPath para incluir el resultado con la entrada
- Se utiliza ResultPath para actualizar un nodo de la entrada con el resultado
- Se utiliza ResultPath para incluir tanto el error como la entrada en un Catch

> **Tip**
>
> Utilice el simulador de flujo de datos de la consola de Step Functions para probar la sintaxis de las rutas JSON, comprender mejor cómo se manipulan los datos dentro de un estado y ver cómo se transfieren los datos entre los estados.

Se usa ResultPath para reemplazar la entrada con el resultado

Si no especifica una `ResultPath`, el comportamiento predeterminado es como si hubiera especificado `"ResultPath": "$"`. Dado que esto indica al estado que reemplace toda la entrada por el resultado, la entrada de estado se reemplaza por completo por el resultado procedente del resultado de la tarea.

En el siguiente diagrama, se muestra cómo `ResultPath` puede reemplazar por completo la entrada por el resultado de la tarea.



Utilice la máquina de estados y la función de Lambda que se describen en Creación de una máquina de estado de Step Functions que utilice Lambda y cambie el tipo de integración de servicios a integración del SDK de AWS para la función de Lambda. Para ello, especifique la función de Lambda Nombre de recurso de Amazon (ARN) en el campo `Resource` del estado `Task`, como se muestra en el siguiente ejemplo. El uso de la integración del AWS SDK garantiza que el resultado del `Task` estado solo contenga el resultado de la función Lambda sin metadatos.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
```

```
}
```

A continuación, pase la siguiente entrada:

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

La función de Lambda proporciona el siguiente resultado:

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> Puede ver este resultado en la consola de Step Functions. Para ello, en la página Detalles de ejecución de la consola, elija la función Lambda en la Vista de gráfico. A continuación, seleccione la pestaña Salida en el panel Detalles del paso para ver este resultado.

Si no se especifica `ResultPath` en el estado o si se ha establecido `"ResultPath": "$"`, la entrada del estado se sustituye por el resultado de la función de Lambda y la salida del estado es la siguiente.

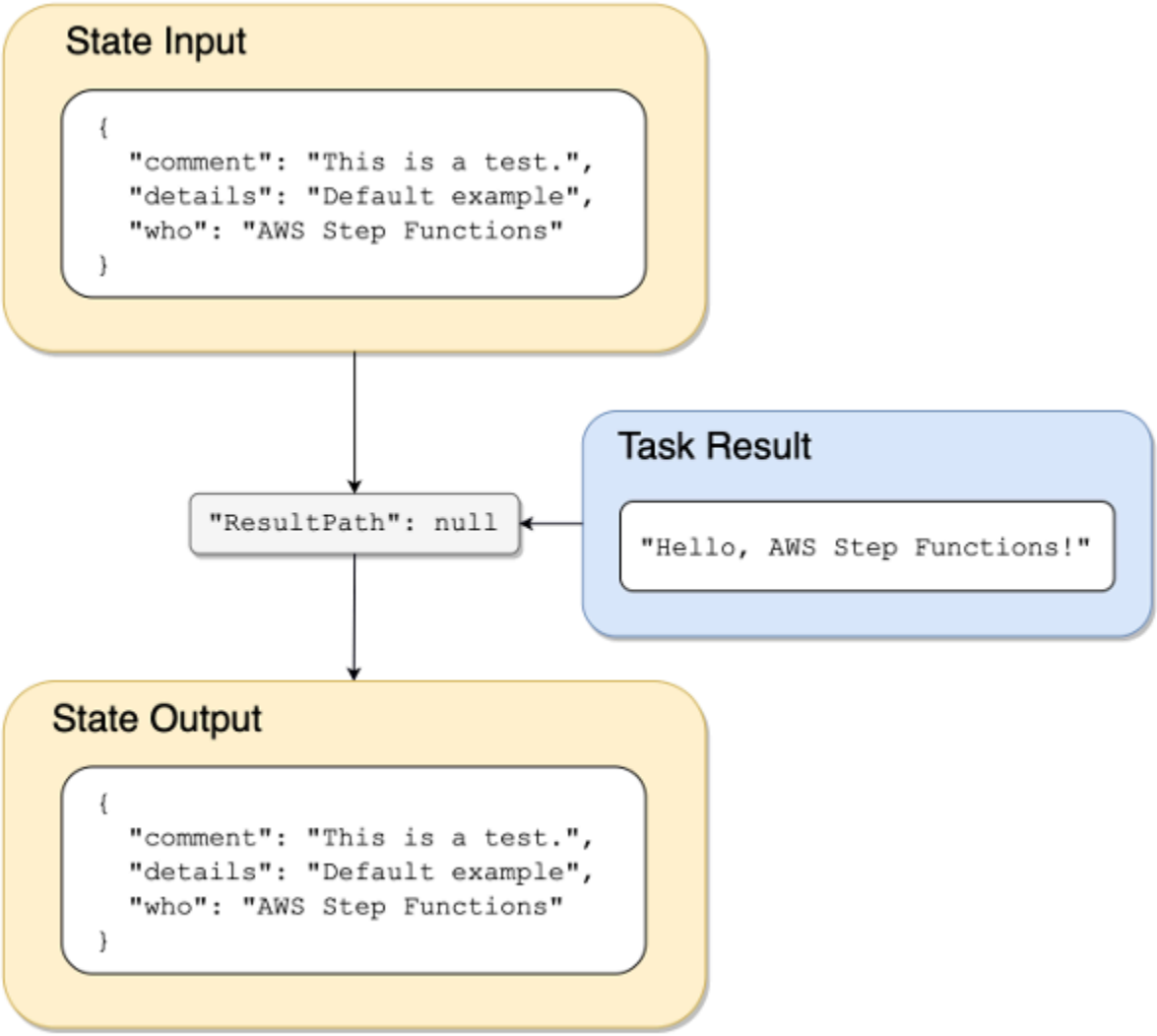
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> `ResultPath` se utiliza para incluir contenido a partir del resultado con la entrada, antes de pasarlo a la salida. Pero, si no se especifica `ResultPath`, el valor predeterminado consiste en reemplazar toda la entrada.

Descartar el resultado y conservar la entrada original

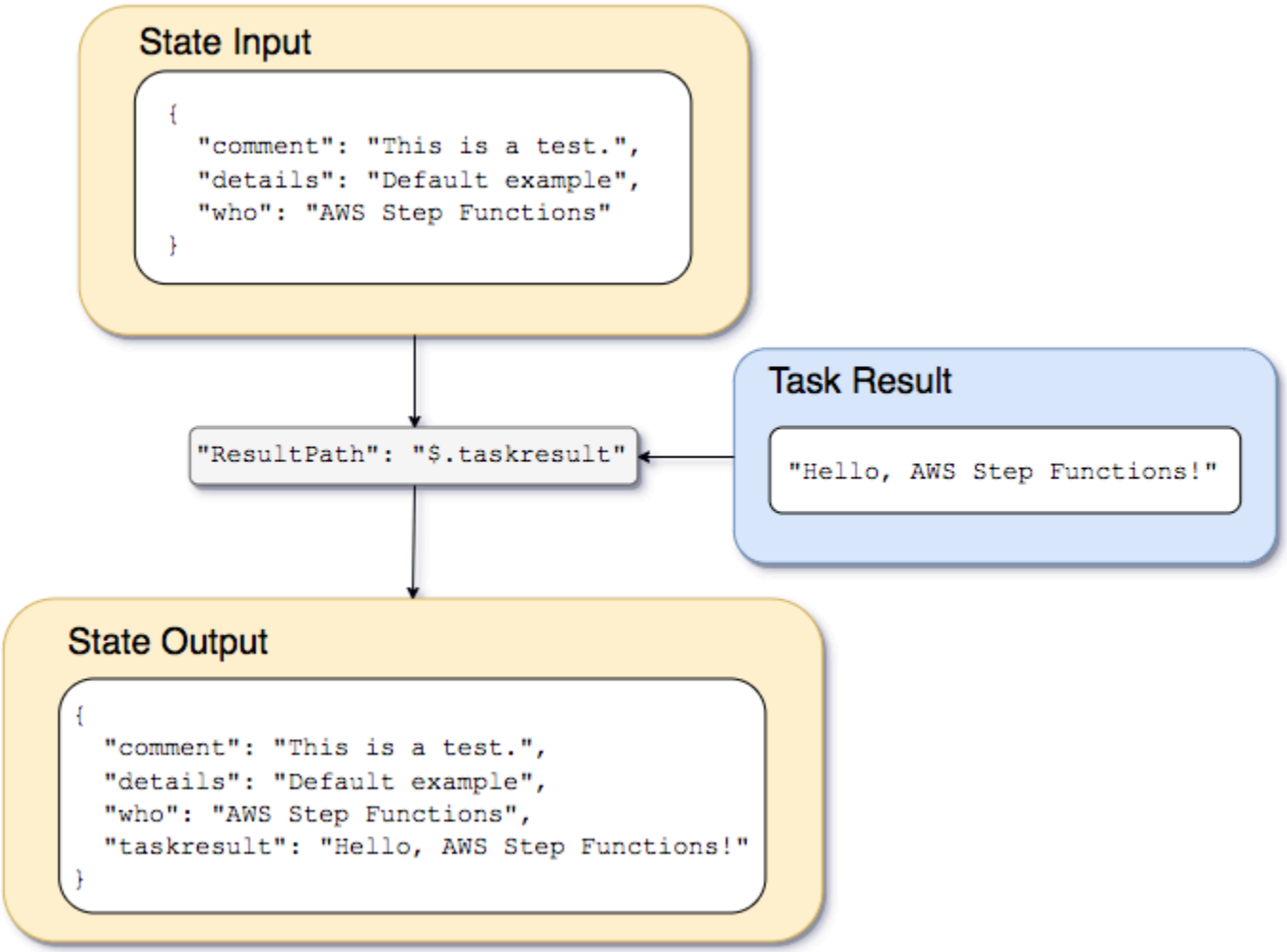
Si establece `ResultPath` en `null`, se transferirá la entrada original a la salida. Al utilizar `"ResultPath": null`, la carga de entrada del estado se copiará directamente en la salida, independientemente del resultado.

El siguiente diagrama muestra cómo una `ResultPath` nula copiará la entrada directamente en la salida.



Se utiliza ResultPath para incluir el resultado con la entrada

El diagrama siguiente muestra cómo `ResultPath` puede incluir el resultado con la entrada.



Con la máquina de estado y la función de Lambda descritas en el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda, podríamos pasar la siguiente entrada.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

El resultado de la función de Lambda es el siguiente.

```
"Hello, AWS Step Functions!"
```

Si se desea conservar la entrada, inserte el resultado de la función de Lambda y, a continuación, pase el JSON combinado al siguiente estado; podríamos establecer `ResultPath` en lo siguiente.

```
"ResultPath": "$.taskresult"
```

Esto incluye el resultado de la función de Lambda con la entrada original.

```
{
  "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!"
}
```

La salida de la función de Lambda se añade a la entrada original como un valor para `taskresult`. La entrada, incluido el valor recién insertado, se pasa al siguiente estado.

También puede insertar el resultado en un nodo secundario de la entrada. Establezca `ResultPath` como se indica a continuación.

```
"ResultPath": "$.strings.lambdaresult"
```

Inicie una ejecución utilizando la siguiente entrada.

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

El resultado de la función de Lambda se inserta como un elemento secundario del nodo `strings` en la entrada:

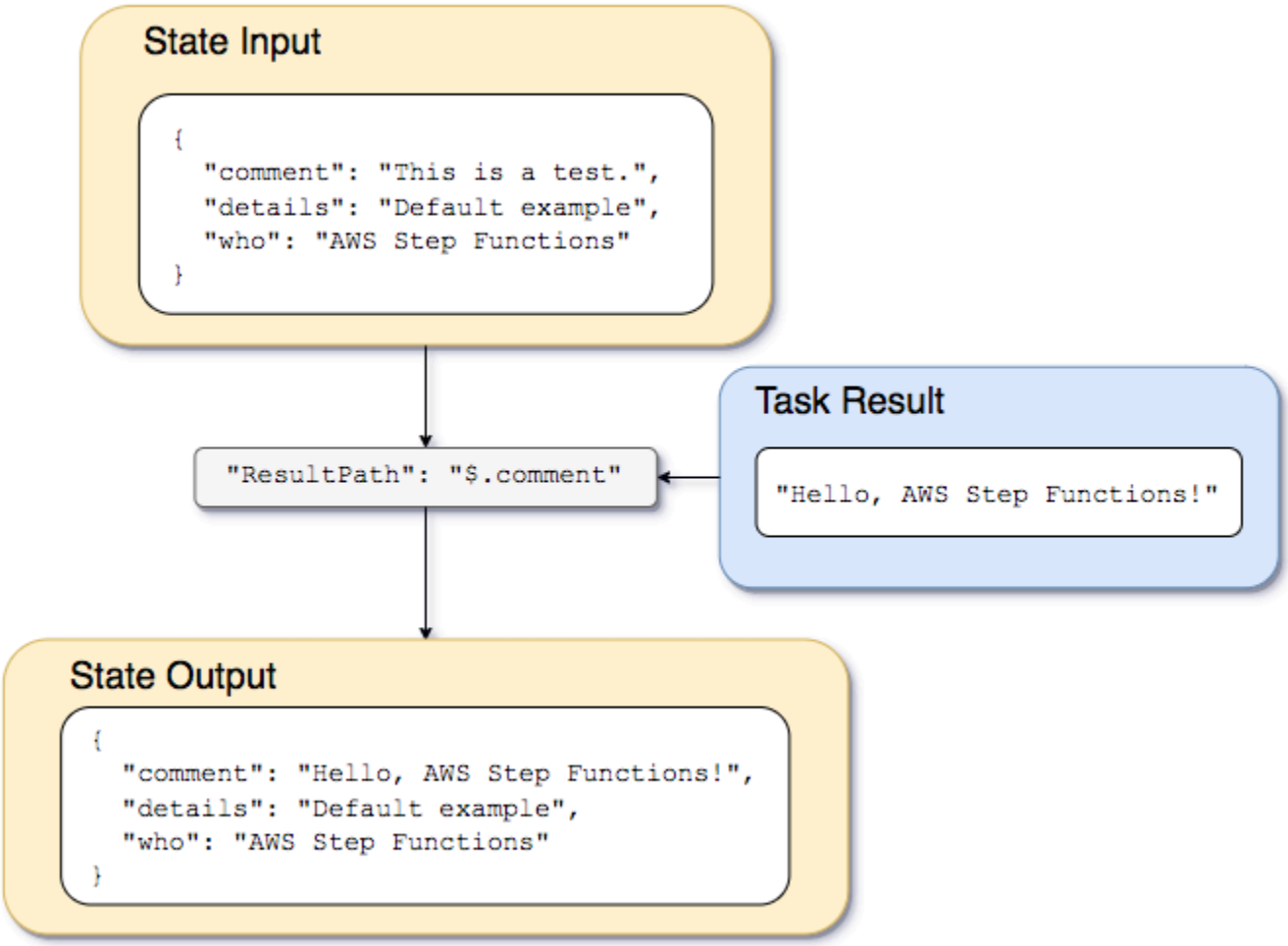
```
{
  "comment": "An input comment.",
```

```
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz",
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

La salida de estado ahora incluye el JSON de entrada original con el resultado como nodo secundario.

Se utiliza ResultPath para actualizar un nodo de la entrada con el resultado

En el siguiente diagrama, se muestra cómo `ResultPath` puede actualizar el valor de los nodos JSON existentes en la entrada por los valores del resultado de tarea.



Con el ejemplo de la máquina de estado y la función de Lambda descritos en el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda, podríamos pasar la siguiente entrada.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

El resultado de la función de Lambda es el siguiente.

```
Hello, AWS Step Functions!
```

En lugar de conservar la entrada e insertar el resultado como un nuevo nodo en el JSON, podemos sobrescribir un nodo existente.

Por ejemplo, del mismo modo que al omitir o establecer `"ResultPath": "$"` se sobrescribe todo el nodo, se puede especificar un nodo determinado para sobrescribirlo con el resultado.

```
"ResultPath": "$.comment"
```

Como el nodo `comment` ya existe en la entrada de estado, al establecer `ResultPath` en `"$.comment"` se reemplaza dicho nodo en la entrada por el resultado de la función de Lambda. Sin filtrar más mediante `OutputPath`, se pasa lo siguiente a la salida.

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

El valor para el nodo `comment`, `"This is a test of the input and output of a Task state."`, se reemplaza por el resultado de la función de Lambda: `"Hello, AWS Step Functions!"` en la salida de estado.

Se utiliza ResultPath para incluir tanto el error como la entrada en un **`Catch`**

El tutorial Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos muestra cómo utilizar una máquina de estado para detectar un error. En algunos casos, es

posible que desee conservar la entrada original con el error. Utilice `ResultPath` en un `Catch` para incluir el error con la entrada original, en lugar de sustituirla.

```
"Catch": [{
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

Si la instrucción `Catch` anterior detecta un error, incluye el resultado en un nodo `error` dentro de la entrada de estado. Por ejemplo, con la siguiente entrada:

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

La salida de estado al detectar un error es la siguiente.

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

Para obtener más información sobre el control de errores, consulte lo siguiente:

- Control de errores en Step Functions
- Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos

OutputPath

`OutputPath` le permite seleccionar una parte de la salida de estado para pasar al siguiente estado. Esto le permite filtrar la información no deseada y pasar solo la parte de JSON que le interese.

Si no especifica una `OutputPath`, el valor predeterminado es `$`. Esto pasa todo el nodo JSON (determinado por la entrada de estado, el resultado de la tarea y `ResultPath`) al siguiente estado.

> **Tip**
>
> Utilice el simulador de flujo de datos de la consola de Step Functions para probar la sintaxis de las rutas JSON, comprender mejor cómo se manipulan los datos dentro de un estado y ver cómo se transfieren los datos entre los estados.

Para más información, consulte los siguientes temas:

- Rutas en Amazon States Language
- Ejemplos de InputPath, ResultPath y OutputPat
- Cómo pasar JSON estático como parámetros
- Procesamiento de entrada y salida en Step Functions

Ejemplos de InputPath, ResultPath y OutputPat

Cualquier estado que no sea un estado Fail o un estado Succeed puede incluir los campos de procesamiento de entrada y salida, tales como `InputPath`, `ResultPath` o `OutputPath`. Además, los estados Wait y Choice no admiten el campo `ResultPath`. Con estos campos, puede usar JSONPath para filtrar los datos JSON a medida que avanzan por el flujo de trabajo.

También puede usar el campo `Parameters` para manipular los datos JSON a medida que avanzan por el flujo de trabajo. Para obtener más información sobre del uso de `Parameters`, consulte InputPath, Parámetros y ResultSelector.

Por ejemplo, comience con la función de AWS Lambda y la máquina de estado que se describen en el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda. Modifique la máquina de estado de forma que incluya el siguiente `InputPath`, `ResultPath` y `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
    }
  }
}
```

Inicie una ejecución utilizando la siguiente entrada.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Supongamos que los nodos `comment` y `extra` se puedan descartar, pero que deseamos incluir la salida de la función de Lambda y conservar la información en el nodo `data`.

En la máquina de estado actualizada, el estado `Task` se modifica para procesar la entrada a la tarea.

```
"InputPath": "$.lambda",
```

Esta línea en la definición de la máquina de estado limita la entrada de tarea a solo el nodo `lambda` de la entrada del estado. La función de Lambda recibe solo el objeto JSON `{"who": "AWS Step Functions"}` como entrada.

```
"ResultPath": "$.data.lambdaresult",
```

Esta `ResultPath` indica a la máquina de estado que inserte el resultado de la función de Lambda en un nodo denominado `lambdaresult`, como elemento secundario del nodo `data` de la entrada de la máquina de estado original. Como no estamos realizando ninguna otra manipulación en la entrada original y en el resultado que utiliza `OutputPath`, la salida del estado ahora incluye el resultado de la función de Lambda con la entrada original.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Sin embargo, nuestro objetivo era conservar únicamente el nodo `data` e incluir el resultado de la función de Lambda. `OutputPath` filtra este JSON combinado antes de pasárselo al resultado del estado.

```
"OutputPath": "$.data",
```

Esto selecciona solo el nodo `data` de la entrada original (incluido el nodo secundario `lambdaresult` insertado por `ResultPath`) para pasarlo a la salida. La salida del estado se filtra para mostrar lo siguiente.

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

En este estado `Task`:

1. `InputPath` envía solo el nodo `lambda` desde la entrada a la función de Lambda.
2. `ResultPath` inserta el resultado como un elemento secundario del nodo `data` de la entrada original.
3. `OutputPath` filtra la entrada del estado (que ahora incluye el resultado de la función de Lambda) de forma que solo transfiere el nodo `data` a la salida del estado.

Example para manipular la entrada, el resultado y la salida final de la máquina de estado original mediante JSONPath

Considere la siguiente máquina de estado que verifica la identidad y la dirección de un solicitante de seguro.

> **Note**
>
> Para ver el ejemplo completo, consulte Cómo usar la ruta JSON in Step Functions.

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

Si ejecuta esta máquina de estado con la siguiente entrada, la ejecución producirá un error, porque las funciones de Lambda que realizan la verificación solo esperan los datos que deben verificarse como entrada. Por lo tanto, debe especificar los nodos que contienen la información que se va a verificar mediante un valor de JSONPath adecuado.

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

Para especificar el nodo que debe usar la función de Lambda *`check-identity`*, utilice el campo `InputPath` de la siguiente manera:

```
"InputPath": "$.data.identity"
```

Y para especificar el nodo que debe utilizar la función de Lambda *`check-address`*, utilice el campo `InputPath` de la siguiente manera:

```
"InputPath": "$.data.address"
```

Ahora, si desea almacenar el resultado de la verificación en la entrada original de la máquina de estado, utilice el campo `ResultPath` de la siguiente manera:

```
"ResultPath": "$.results"
```

Sin embargo, si solo necesita los resultados de identidad y verificación y descarta la entrada original, utilice el campo `OutputPath` de la siguiente manera:

```
"OutputPath": "$.results"
```

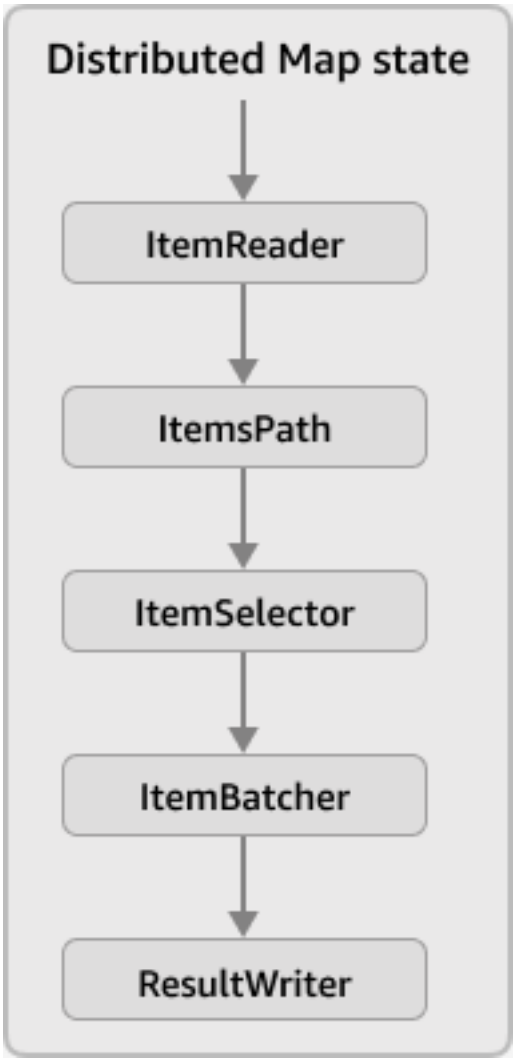
Para obtener más información, consulte Procesamiento de entrada y salida en Step Functions.

Campos de entrada y salida del estado Map

Los estados `Map` iteran simultáneamente sobre una colección de elementos de un conjunto de datos, como una matriz JSON, una lista de objetos de Amazon S3 o las filas de un archivo CSV en un bucket de Amazon S3. Repite un conjunto de pasos para cada elemento de la colección. Puede configurar la entrada que recibe el estado `Map` y la salida que genera mediante estos campos. Step Functions aplica cada campo del estado Map Distributed en el orden que se muestra en las siguientes lista e ilustración:

> **Note**
>
> Según el caso de uso, es posible que no necesite aplicar todos estos campos.

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher
5. ResultWriter



> **Note**
>
> Estos campos de entrada y salida del estado `Map` no están disponibles actualmente en el simulador de flujo de datos de la consola de Step Functions.

ItemReader

El campo `ItemReader` es un objeto JSON que especifica un conjunto de datos y su ubicación. Un estado Map Distributed usa este conjunto de datos como entrada. En el siguiente ejemplo, se muestra la sintaxis del campo `ItemReader` si el conjunto de datos es un archivo CSV almacenado en un bucket de Amazon S3.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> En Workflow Studio, se especifica el conjunto de datos y su ubicación en el campo Fuente del elemento.

Contenido

- Contenido del campo ItemReader
- Ejemplos de conjuntos de datos
- Políticas de IAM para conjuntos de datos

Contenido del campo ItemReader

El contenido del campo `ItemReader` varía según el conjunto de datos. Por ejemplo, si el conjunto de datos es una matriz JSON transferida desde un paso anterior del flujo de trabajo, el campo `ItemReader` se omite. Si el conjunto de datos es un origen de datos de Amazon S3, este campo contiene los siguientes subcampos.

`ReaderConfig`

Un objeto JSON que especifica los detalles siguientes:

- `InputType`

  Especifica el tipo de origen de datos de Amazon S3, como un archivo CSV, un objeto, un archivo JSON o una lista de inventario de Amazon S3. En Workflow Studio, puede seleccionar un tipo de entrada de la lista desplegable Origen del elemento de Amazon S3 en el campo Fuente del elemento.

- `CSVHeaderLocation`

  > **Note**
  >
  > Debe especificar este campo solo si utiliza un archivo CSV como conjunto de datos.

  Acepta uno de los siguientes valores para especificar la ubicación del encabezado de la columna:

  > **Important**
  >
  > Actualmente, Step Functions admite encabezados CSV de hasta 10 KB.

  - `FIRST_ROW` – Utilice esta opción si la primera línea del archivo es el encabezado.
  - `GIVEN` – Utilice esta opción para especificar el encabezado dentro de la definición de la máquina de estado. Por ejemplo, si el archivo CSV contiene los datos siguientes.

    ```
    1,307,3.5,1256677221
    1,481,3.5,1256677456
    1,1091,1.5,1256677471
    ```

```
...
```

Proporcione la siguiente matriz JSON como encabezado CSV.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> En Workflow Studio, puede encontrar esta opción en Configuración adicional, en el campo Fuente del elemento.

- `MaxItems`

  Limita el número de elementos de datos que se pasan al estado `Map`. Por ejemplo, supongamos que proporciona un archivo CSV que contiene 1000 filas y especifica un límite de 100. Entonces, el intérprete pasa solo 100 filas al estado `Map`. El estado `Map` procesa los elementos en orden secuencial, a partir de la fila siguiente al encabezado.

  De forma predeterminada, el estado `Map` se repite en todos los elementos del conjunto de datos especificado.

  > **Note**
  >
  > Actualmente, puede especificar un límite de hasta 100 000 000. El estado Map Distributed deja de leer los elementos que superen este límite.

> **Tip**
>
> En Workflow Studio, puede encontrar esta opción en Configuración adicional, en el campo Fuente del elemento.

También puede especificar una ruta de referencia a un par clave-valor existente en la entrada del estado Map Distributed. Esta ruta debe convertirse en un número entero positivo. La ruta de referencia se especifica en el subcampo `MaxItemsPath`.

> **Important**
>
> Puede especificar el subcampo `MaxItems` o `MaxItemsPath`, pero no ambos.

`Resource`

La acción de la API de Amazon S3 que Step Functions debe invocar en función del conjunto de datos especificado.

`Parameters`

Un objeto JSON que especifica el nombre del bucket de Amazon S3 y la clave de objeto en los que se almacena el conjunto de datos.

> **Important**
>
> Asegúrese de que los buckets de Amazon S3 estén en las mismas Cuenta de AWS y Región de AWS que la máquina de estado.

Ejemplos de conjuntos de datos

Puede especificar una de las opciones siguientes como conjunto de datos:

- Matriz JSON de un paso anterior
- Lista de objetos de Amazon S3
- Archivo JSON en un bucket de Amazon S3

- Archivo CSV en un bucket de Amazon S3
- Lista de inventario de Amazon S3

> **Important**
>
> Step Functions necesita los permisos adecuados para obtener acceso a los conjuntos de datos de Amazon S3 que utilice. Para obtener información sobre las políticas de IAM para los conjuntos de datos, consulte Políticas de IAM para conjuntos de datos.

Matriz JSON de un paso anterior

Un estado Map Distributed puede aceptar una entrada JSON transferida desde un paso anterior del flujo de trabajo. Esta entrada debe ser una matriz o debe contener una matriz dentro de un nodo específico. Para seleccionar un nodo que contenga la matriz, puede usar el campo `ItemsPath`.

Para procesar los elementos individuales de la matriz, el estado Map Distributed inicia la ejecución de un flujo de trabajo secundario para cada elemento de la matriz. Las siguientes pestañas muestran ejemplos de la entrada que se transfiere al estado `Map` y la entrada correspondiente a la ejecución de un flujo de trabajo secundario.

> **Note**
>
> Step Functions omite el campo `ItemReader` cuando el conjunto de datos es una matriz JSON de un paso anterior.

Input passed to the Map state

Considera la siguiente matriz JSON de tres elementos.

```
"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

El estado Map Distributed inicia tres ejecuciones de flujos de trabajo secundarios. Cada ejecución recibe un elemento de matriz como entrada. El siguiente ejemplo muestra la entrada recibida por la ejecución de un flujo de trabajo secundario.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Ejemplo de objetos de Amazon S3

Un estado Map Distributed puede iterar sobre los objetos que se almacenan en un bucket de Amazon S3. Cuando la ejecución del flujo de trabajo alcanza el estado `Map`, Step Functions invoca la acción de la API ListObjectsV2, que devuelve una matriz de metadatos de objetos de Amazon S3. En esta matriz, cada elemento contiene datos, como ETag y Key, de los datos almacenados en el bucket.

Para procesar los elementos individuales de la matriz, el estado Map Distributed inicia la ejecución de un flujo de trabajo secundario. Por ejemplo, suponga que su bucket de Amazon S3 contiene 100 imágenes. A continuación, la matriz devuelta tras invocar la acción de la API `ListObjectsV2` contiene 100 elementos. A continuación, el estado Map Distributed inicia 100 ejecuciones de flujos de trabajo secundarios para procesar cada elemento de la matriz.

> **Note**
>
> - En la actualidad, Step Functions también incluye un elemento para cada carpeta que cree en un bucket de Amazon S3 específico utilizando la consola de Amazon S3. Esto se

traduce en una ejecución de flujo de trabajo secundaria adicional iniciada por el estado Map Distributed. Para evitar crear una ejecución de flujo de trabajo secundaria adicional para la carpeta, le recomendamos que la utilice AWS CLI para crear carpetas. Para obtener más información, consulte Uso de comandos de S3 de alto nivel en la Guía del usuario de AWS Command Line Interface.

- Step Functions necesita los permisos adecuados para obtener acceso a los conjuntos de datos de Amazon S3 que utilice. Para obtener información sobre las políticas de IAM para los conjuntos de datos, consulte Políticas de IAM para conjuntos de datos.

En las siguientes pestañas se muestran ejemplos de la sintaxis del campo `ItemReader` y de la entrada que se transfiere a la ejecución de un flujo de trabajo secundario para este conjunto de datos.

ItemReader syntax

En este ejemplo, ha organizado los datos, que incluyen imágenes, archivos JSON y objetos, dentro de un prefijo llamado `processData` en un bucket de Amazon S3 llamado `myBucket`.

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

El estado Map Distributed inicia tantas ejecuciones de flujos de trabajo secundarios como el número de elementos presentes en el bucket de Amazon S3. El siguiente ejemplo muestra la entrada recibida por la ejecución de un flujo de trabajo secundario.

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

Archivo JSON en un bucket de Amazon S3

Un estado Map Distributed puede aceptar un archivo JSON almacenado en un bucket de Amazon S3 como conjunto de datos. El archivo JSON debe contener una matriz.

Cuando la ejecución del flujo de trabajo alcanza el estado `Map`, Step Functions invoca la acción de la API GetObject para recuperar el archivo JSON especificado. A continuación, el estado `Map` se repite sobre cada elemento de la matriz e inicia la ejecución de un flujo de trabajo secundario para cada elemento. Por ejemplo, si el archivo JSON contiene 1000 elementos de matriz, el estado `Map` inicia 1000 ejecuciones de flujos de trabajo secundarios.

> **Note**
>
> - La entrada de ejecución utilizada para iniciar la ejecución de un flujo de trabajo secundario no puede superar los 256 KB. Sin embargo, Step Functions permite leer un elemento de hasta 8 MB de un archivo CSV o JSON si, a continuación, se aplica el campo `ItemSelector` opcional para reducir el tamaño del elemento.
> - Actualmente, Step Functions admite 10 GB como tamaño máximo de un archivo individual en un informe de inventario de Amazon S3. Sin embargo, Step Functions puede procesar más de 10 GB si cada archivo individual tiene menos de 10 GB.
> - Step Functions necesita los permisos adecuados para obtener acceso a los conjuntos de datos de Amazon S3 que utilice. Para obtener información sobre las políticas de IAM para los conjuntos de datos, consulte Políticas de IAM para conjuntos de datos.

En las siguientes pestañas se muestran ejemplos de la sintaxis del campo `ItemReader` y de la entrada que se transfiere a la ejecución de un flujo de trabajo secundario para este conjunto de datos.

Para este ejemplo, imagine que tiene un archivo JSON llamado *`factcheck.json`*. Ha almacenado este archivo en un prefijo llamado *`jsonDataset`* en un bucket de Amazon S3. A continuación se muestra un ejemplo de conjunto de datos JSON.

```
[
  {
    "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

El estado Map Distributed inicia tantas ejecuciones de flujos de trabajo secundarios como el número de elementos de la matriz presentes en el archivo JSON. El siguiente ejemplo muestra la entrada recibida por la ejecución de un flujo de trabajo secundario.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Archivo CSV en un bucket de Amazon S3

Un estado Map Distributed puede aceptar un archivo CSV almacenado en un bucket de Amazon S3 como conjunto de datos. Si usa un archivo CSV como conjunto de datos, debe especificar un encabezado de columna CSV. Para obtener información sobre cómo especificar un prefijo personalizado, consulte Contenido del campo ItemReader.

Puesto que no existe un formato estandarizado para crear y mantener los datos en los archivos CSV, Step Functions analiza los archivos CSV según las siguientes reglas:

- Las comas (,) son un delimitador que separa los campos individuales.
- Los retornos de carro son un delimitador que separa los registros individuales.
- Los campos se tratan como cadenas. Para las conversiones de tipos de datos, utilice la función intrínseca `States.StringToJson` en ItemSelector.
- No es necesario incluir comillas dobles (" ") para delimitar cadenas. No obstante, las cadenas entre comillas dobles pueden contener comas y retornos de carro sin que funcionen como delimitadores.
- Para incluir las comillas dobles en una secuencia de escape, repítalas.
- Si el número de campos de una fila es inferior al número de campos del encabezado, Step Functions proporciona cadenas vacías para los valores que faltan.
- Si el número de campos de una fila es mayor que el número de campos del encabezado, Step Functions omite los campos adicionales.

Para obtener más información acerca de cómo Step Functions analiza un archivo CSV, consulte Example of parsing an input CSV file.

Cuando la ejecución del flujo de trabajo alcanza el estado `Map`, Step Functions invoca la acción de la API GetObject para recuperar el archivo CSV especificado. A continuación, el estado `Map` se repite en cada fila del archivo CSV e inicia la ejecución de un flujo de trabajo secundario para procesar los elementos de cada fila. Por ejemplo, suponga que proporciona un archivo CSV que contiene 100 filas como entrada. Entonces, el intérprete pasa cada fila al estado `Map`. El estado `Map` procesa los elementos en orden de serie, a partir de la fila siguiente al encabezado.

> **Note**
>
> - La entrada de ejecución utilizada para iniciar la ejecución de un flujo de trabajo secundario no puede superar los 256 KB. Sin embargo, Step Functions permite leer un elemento

de hasta 8 MB de un archivo CSV o JSON si, a continuación, se aplica el campo `ItemSelector` opcional para reducir el tamaño del elemento.

- Actualmente, Step Functions admite 10 GB como tamaño máximo de un archivo individual en un informe de inventario de Amazon S3. Sin embargo, Step Functions puede procesar más de 10 GB si cada archivo individual tiene menos de 10 GB.
- Step Functions necesita los permisos adecuados para obtener acceso a los conjuntos de datos de Amazon S3 que utilice. Para obtener información sobre las políticas de IAM para los conjuntos de datos, consulte Políticas de IAM para conjuntos de datos.

En las siguientes pestañas se muestran ejemplos de la sintaxis del campo `ItemReader` y de la entrada que se transfiere a la ejecución de un flujo de trabajo secundario para este conjunto de datos.

ItemReader syntax

Por ejemplo, supongamos que tiene un archivo CSV llamado *`ratings.csv`*. A continuación, ha almacenado este archivo dentro de un prefijo llamado *`csvDataset`* en un bucket de Amazon S3.

```
{
  "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

El estado Map Distributed inicia tantas ejecuciones de flujos de trabajo secundarios como el número de filas presentes en el archivo CSV, excluida la fila del encabezado, si está en el archivo. El siguiente ejemplo muestra la entrada recibida por la ejecución de un flujo de trabajo secundario.

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
  "timestamp": "1256677221"
}
```

Ejemplo de inventario de S3

Un estado Map Distributed puede aceptar un archivo de manifiesto de inventario de Amazon S3 almacenado en un bucket de Amazon S3 como conjunto de datos.

Cuando la ejecución del flujo de trabajo alcanza el estado `Map`, Step Functions invoca la acción de la API GetObject para recuperar el archivo de manifiesto de inventario de Amazon S3 especificado. A continuación, el estado `Map` recorre en iteración los objetos del inventario para devolver una matriz de metadatos de objetos de inventario de Amazon S3.

> **Note**
>
> - Actualmente, Step Functions admite 10 GB como tamaño máximo de un archivo individual en un informe de inventario de Amazon S3. Sin embargo, Step Functions puede procesar más de 10 GB si cada archivo individual tiene menos de 10 GB.
> - Step Functions necesita los permisos adecuados para obtener acceso a los conjuntos de datos de Amazon S3 que utilice. Para obtener información sobre las políticas de IAM para los conjuntos de datos, consulte Políticas de IAM para conjuntos de datos.

A continuación se muestra un ejemplo de un archivo de inventario de ejemplo en formato CSV: Este archivo incluye los objetos denominados `csvDataset` `yimageDataset`, que se almacenan en un bucket de Amazon S3 que lleva ese nombre `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**
>
> Actualmente, Step Functions no admite el informe de inventario de Amazon S3 definido por el usuario como conjunto de datos. También debe asegurarse de que el formato de salida del informe de inventario de Amazon S3 sea CSV. Para obtener más información sobre los inventarios de Amazon S3 y cómo configurarlos, consulte Inventario de Amazon S3 en la Guía del usuario de Amazon S3.

El siguiente ejemplo de un archivo de manifiesto de inventario muestra los encabezados CSV de los metadatos de los objetos de inventario.

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

En las siguientes pestañas se muestran ejemplos de la sintaxis del campo `ItemReader` y de la entrada que se transfiere a la ejecución de un flujo de trabajo secundario para este conjunto de datos.

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"
}
```

En función de los campos que haya seleccionado al configurar el informe de inventario de Amazon S3, el contenido del archivo `manifest.json` puede variar respecto al ejemplo mostrado.

Políticas de IAM para conjuntos de datos

Al crear flujos de trabajo con la consola de Step Functions, Step Functions puede generar automáticamente políticas de IAM basadas en los recursos de la definición de flujo de trabajo. Estas políticas incluyen los privilegios mínimos necesarios para permitir que el rol de la máquina de estado invoque la acción de la API `StartExecution` para el estado Map Distributed. Estas políticas también incluyen los privilegios mínimos necesarios para que Step Functions pueda acceder a los recursos de AWS, como buckets y objetos de Amazon S3 y las funciones de Lambda. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM. Por ejemplo, si el flujo de trabajo incluye un estado `Map` en modo distribuido, aplique las políticas al bucket y a la carpeta de Amazon S3 específicos que contengan el conjunto de datos.

> **Important**
>
> Si especifica un bucket y un objeto de Amazon S3, o un prefijo, con una ruta de referencia a un par clave-valor existente en la entrada del estado Map Distributed, no olvide actualizar las políticas de IAM para el flujo de trabajo. Limite las políticas hasta los nombres de objeto y bucket a los que se dirige la ruta en tiempo de ejecución.

Los siguientes ejemplos de políticas de IAM otorgan los privilegios mínimos necesarios para acceder a los conjuntos de datos de Amazon S3 mediante las acciones de la API ListObjectsV2 y GetObject.

Example Política de IAM para el objeto Amazon S3 como conjunto de datos

El siguiente ejemplo muestra una política de IAM que concede los privilegios mínimos necesarios para acceder a los objetos organizados dentro de *processImages* en un bucket de Amazon S3 llamado *myBucket*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example Política de IAM para un archivo CSV como conjunto de datos

En el siguiente ejemplo se muestra una política de IAM que concede los privilegios mínimos necesarios para acceder a un archivo CSV llamado *ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Política de IAM para un inventario de Amazon S3 como conjunto de datos

En el siguiente ejemplo se muestra una política de IAM que concede los privilegios mínimos necesarios para acceder a un informe de inventario de Amazon S3.

```
{
    "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

Utilice el campo `ItemsPath` para seleccionar una matriz dentro de una entrada JSON proporcionada a un estado `Map`. El estado `Map` repite un conjunto de pasos para cada elemento de la matriz. De forma predeterminada, un estado `Map` establece `ItemsPath` en `$`, lo que selecciona toda la entrada. Si la entrada del estado `Map` es una matriz JSON, ejecutará una iteración para cada elemento de la matriz y pasará ese elemento a la iteración como entrada.

> **Note**
>
> Puede usar `ItemsPath` en el estado Map Distributed solo si usa una entrada JSON pasada desde un estado anterior del flujo de trabajo.

Puede usar el campo `ItemsPath` para especificar una ubicación en la entrada que apunte a la matriz JSON utilizada en las iteraciones. El valor de `ItemsPath` debe ser una ruta de referencia y esa ruta debe apuntar a una matriz JSON. Por ejemplo, supongamos que la entrada a un estado `Map` incluye dos matrices, como en el siguiente ejemplo.

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
  "ThingsGiantsSay": [
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
    },
    {
      "say": "Fo!"
    },
    {
      "say": "Fum!"
    }
  ]
}
```

En este caso, puede especificar qué matriz se ha de utilizar para las iteraciones del estado `Map` seleccionándola mediante `ItemsPath`. La siguiente definición de máquina de estado especifica

la matriz `ThingsPiratesSay` en la entrada mediante `ItemsPath`. A continuación, ejecuta una iteración del estado de paso `SayWord` para cada elemento de la matriz `ThingsPiratesSay`.

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```

Al procesar la entrada, el `Map` estado aplica `ItemsPath` después de `InputPath`. Opera en la entrada efectiva al estado, después de que `InputPath` filtre la entrada.

Para obtener más información acerca de los estados `Map`, consulte los temas siguientes:

- Estado Map
- Modos de procesamiento del estado Map
- Repetir una acción utilizando el estado Inline Map
- Procesamiento de entrada y salida del estado `Map` en línea

ItemSelector

De forma predeterminada, la entrada efectiva para el estado `Map` es el conjunto de elementos de datos individuales presentes en la entrada de estado sin procesar. El campo `ItemSelector` permite anular los valores de los elementos de datos antes de que se transfieran al estado `Map`. Para anular los valores, especifique una entrada JSON válida que contenga un conjunto de pares clave-valor. Estos pares pueden ser valores estáticos proporcionados en la definición de la máquina de estado,

valores seleccionados de la entrada de estado mediante una ruta o valores a los que se acceda desde el objeto de contexto.

Si especifica pares clave-valor mediante una ruta o un objeto de contexto, el nombre de la clave debe terminar en `.$`.

> **Note**
>
> El campo `ItemSelector` reemplaza al campo `Parameters` dentro del estado `Map`. Si utiliza el campo `Parameters` en las definiciones de estado `Map` para crear entradas personalizadas, le recomendamos encarecidamente que lo sustituya por `ItemSelector`.

Puede especificar el campo `ItemSelector` tanto en un estado Map en línea como en un estado Map Distributed.

Por ejemplo, considere la siguiente entrada JSON que contiene una matriz de tres elementos dentro del nodo `imageData`. Por cada iteración del estado *`Map`*, se pasa un elemento de matriz a la iteración como entrada.

```
[
  {
    "resize": "true",
    "format": "jpg"
  },
  {
    "resize": "false",
    "format": "png"
  },
  {
    "resize": "true",
    "format": "jpg"
  }
]
```

Con el campo `ItemSelector`, puede definir una entrada JSON personalizada para anular la entrada original, como se muestra en el siguiente ejemplo. Luego, Step Functions pasa esta entrada personalizada a cada iteración del estado *`Map`*. La entrada personalizada contiene un valor estático para `size` y el valor de los datos de un objeto de contexto para el estado `Map`. El objeto de contexto `$$.Map.Item.Value` contiene el valor de cada elemento de datos individual.

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
}
```

El siguiente ejemplo muestra la entrada recibida por una iteración del estado Map en línea:

```
{
  "size": 10,
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> Para ver un ejemplo completo de un estado Map Distributed que usa el campo `ItemSelector`, consulte Introducción al uso del estado Distributed Map.

ItemBatcher

El campo `ItemBatcher` es un objeto JSON que especifica el procesamiento de un grupo de elementos en una sola ejecución de flujo de trabajo secundario. Utilice el procesamiento por lotes cuando procese archivos CSV o matrices JSON de gran tamaño, o conjuntos grandes de objetos de Amazon S3.

En el ejemplo siguiente se muestra la sintaxis del campo `ItemBatcher`. En la siguiente sintaxis, el número máximo de elementos que debe procesar cada ejecución de flujo de trabajo secundario se establece en 100.

```
{
  "ItemBatcher": {
    "MaxItemsPerBatch": 100
  }
}
```

De forma predeterminada, cada elemento de un conjunto de datos se pasa como entrada a las ejecuciones individuales de los flujos de trabajo secundarios. Por ejemplo, supongamos que especifica un archivo JSON como entrada que contiene la siguiente matriz:

```
[
  {
    "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"
  },
  ...
]
```

Para la entrada dada, cada ejecución del flujo de trabajo secundario recibe un elemento de matriz como entrada. En el siguiente ejemplo, se muestra la entrada de la ejecución de un flujo de trabajo secundario:

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Para ayudar a optimizar el rendimiento y el coste del trabajo de procesamiento, seleccione un tamaño de lote que equilibre la cantidad de elementos con el tiempo de procesamiento de los elementos. Si utiliza el procesamiento por lotes, Step Functions añade los elementos a una matriz de elementos. A continuación, pasa la matriz como entrada a cada ejecución del flujo de trabajo secundario. En el siguiente ejemplo se muestra un lote de dos elementos que se pasan como entrada a la ejecución de un flujo de trabajo secundario:

```
{
```

```
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    }
  ]
}
```

Tip

Para obtener más información sobre el uso del campo `ItemBatcher` en los flujos de trabajo, pruebe los siguientes tutoriales y talleres:

- Procesamiento de un lote completo de datos dentro de una función de Lambda
- Realizar iteraciones sobre los elementos de un lote dentro de las ejecuciones de flujos de trabajo secundarios
- Paralelización a gran escala con un estado Map Distributed en el Módulo 14: Procesamiento de datos de El workshop de AWS Step Functions

Contenido

Campos para especificar el procesamiento por lotes de elementos

Para agrupar elementos, especifique el número máximo de elementos a agrupar, el tamaño máximo del lote o ambos. Debe especificar uno de estos valores para agrupar los elementos.

Número máximo de elementos por lote

Especifica el número máximo de elementos que procesa cada ejecución de flujo de trabajo secundario. El intérprete limita el número de elementos agrupados en la matriz `Items` a este

valor. Si se especifica un número y un tamaño de lote, el intérprete reduce el número de elementos de un lote para evitar superar el límite de tamaño de lote especificado.

Si no se especifica este valor, pero se proporciona un valor para el tamaño máximo del lote, Step Functions procesa tantos elementos como sea posible en cada ejecución del flujo de trabajo secundario sin superar el tamaño máximo del lote en bytes.

Por ejemplo, imagine que realiza una ejecución con un archivo JSON de entrada que contiene 1130 nodos. Si se especifica un valor máximo de elementos para cada lote de 100, Step Functions crea 12 lotes. De estos, 11 lotes contienen 100 elementos cada uno, mientras que el duodécimo lote contiene los 30 elementos restantes.

También puede especificar el número máximo de elementos para cada lote como una ruta de referencia a un par clave-valor existente en la entrada de estado Map Distributed. Esta ruta debe convertirse en un número entero positivo.

Por ejemplo, en el caso de la entrada siguiente:

```
{
  "maxBatchItems": 500
}
```

Puede especificar el número máximo de elementos que va a agrupar de la siguiente manera:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Puede especificar el subcampo `MaxItemsPerBatch` o `MaxItemsPerBatchPath`, pero no ambos.

Número máximo de KB por lote

Especifica el tamaño máximo de un lote en bytes, hasta 256 KB. Si se especifica un número y un tamaño de lote máximos, Step Functions reduce el número de elementos de un lote para evitar superar el límite de tamaño de lote especificado.

También puede especificar el número máximo de elementos para cada lote como una ruta de referencia a un par clave-valor existente en la entrada de estado Map Distributed. Esta ruta debe convertirse en un número entero positivo.

> **Note**
>
> Si se utiliza el procesamiento por lotes y no se especifica un tamaño de lote máximo, el intérprete procesa tantos elementos como pueda procesar (hasta 256 KB) en cada ejecución del flujo de trabajo secundario.

Por ejemplo, en el caso de la entrada siguiente:

```
{
  "batchSize": 131072
}
```

Puede especificar el tamaño máximo del lote mediante una ruta de referencia de la siguiente manera:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
```

```
    }
    ...
    ...
  }
}
```

> **Important**
>
> Puede especificar el subcampo `MaxInputBytesPerBatch` o `MaxInputBytesPerBatchPath`, pero no ambos.

Entrada por lotes

Si lo desea, también puede especificar una entrada JSON fija para incluirla en cada lote que se pase a la ejecución de cada flujo de trabajo secundario. Step Functions fusiona esta entrada con la entrada de cada ejecución individual del flujo de trabajo secundario. Por ejemplo, dada la siguiente entrada fija de una fecha de verificación de datos en una serie de elementos:

```
"ItemBatcher": {
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

Cada ejecución de un flujo de trabajo secundario recibe lo siguiente como entrada:

```
{
  "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

El campo `ResultWriter` es un objeto JSON que especifica la ubicación de Amazon S3 en la que Step Functions escribe los resultados de las ejecuciones de flujos de trabajo secundarios iniciadas por un estado Map Distributed. De forma predeterminada, Step Functions no exporta estos resultados.

> **Important**
>
> Asegúrese de que el bucket de Amazon S3 que utilice para exportar los resultados de Map Run esté en la misma Cuenta de AWS y Región de AWS que su máquina de estado. De lo contrario, la ejecución de la máquina de estado fallará y se producirá el error `States.ResultWriterFailed`.

La exportación de los resultados a un bucket de Amazon S3 es útil si el tamaño de la carga de salida es superior a 256 KB. Step Functions consolida todos los datos de ejecución del flujo de trabajo secundario, como la entrada y salida de la ejecución, el ARN y el estado de ejecución. A continuación, exporta las ejecuciones con el mismo estado a sus archivos respectivos en la ubicación de Amazon S3 especificada. En el siguiente ejemplo se muestra la sintaxis del campo `ResultWriter` si se exportan los resultados de la ejecución del flujo de trabajo secundario. En este ejemplo, los resultados se almacenan en un bucket llamado `myOutputBucket` dentro de un prefijo llamado `csvProcessJobs`.

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

> **Tip**
>
> En Workflow Studio, puede exportar los resultados de la ejecución del flujo de trabajo secundario seleccionando Exportar los resultados del estado Map a Amazon S3. A continuación, escriba el nombre del bucket de Amazon S3 y el prefijo donde desea exportar los resultados.

Step Functions necesita los permisos adecuados para acceder al bucket y a la carpeta a los que desea exportar los resultados. Para obtener información acerca de la política de IAM necesarias, consulte Políticas de IAM para ResultWriter.

Si exporta los resultados de la ejecución del flujo de trabajo secundario, la ejecución del estado Map Distributed devuelve el ARN del Map Run y los datos sobre la ubicación de exportación de Amazon S3 en el siguiente formato:

```
{
  "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 exporta las ejecuciones con el mismo estado a sus respectivos archivos. Por ejemplo, si las ejecuciones de flujos de trabajo secundarios dieron como resultado 500 resultados satisfactorios y 200 con error, Step Functions creará dos archivos en la ubicación de Amazon S3 especificada para los resultados correctos y erróneos. En este ejemplo, el archivo de resultados con éxito contiene los 500 resultados con éxito, mientras que el archivo de resultados con error contiene los 200 resultados con error.

Para un intento de ejecución determinado, Step Functions crea los siguientes archivos en la ubicación de Amazon S3 especificada en función del resultado de la ejecución:

- `manifest.json` – Contiene metadatos de Map Run, como la ubicación de exportación, el ARN del Map Run e información sobre los archivos de resultados.

  Si ha utilizado redriven con un Map Run, el archivo `manifest.json` contendrá referencias a todas las ejecuciones correctas de flujos de trabajo secundarios en todos los intentos de un Map

Run. Sin embargo, este archivo contiene referencias a las ejecuciones con errores y pendientes de un uso específico de redrive.

- `SUCCEEDED_n.json` – Contiene los datos consolidados de todas las ejecuciones correctas de flujos de trabajo secundarios. n representa el número de índice del archivo. El número de índice comienza en 0. Por ejemplo, `SUCCEEDED_1.json`.
- `FAILED_n.json` – Contiene los datos consolidados de todas las ejecuciones de flujos de trabajo secundarios con error, con tiempo de espera agotado o anuladas. Utilice este archivo para recuperarse de ejecuciones con error. n representa el índice del archivo. El número de índice comienza en 0. Por ejemplo, `FAILED_1.json`.
- `PENDING_n.json` – Contiene los datos consolidados de todas las ejecuciones de flujos de trabajo secundarios que no se iniciaron porque el Map Run produjo un error o se anuló. n representa el índice del archivo. El número de índice comienza en 0. Por ejemplo, `PENDING_1.json`.

Step Functions admite archivos de resultados individuales de hasta 5 GB. Si el tamaño de un archivo supera los 5 GB, Step Functions crea otro archivo para escribir el resto de los resultados de la ejecución y añade un número de índice al nombre del archivo. Por ejemplo, si el tamaño del archivo `Succeeded_0.json` supera los 5 GB, Step Functions crea un archivo `Succeeded_1.json` para registrar los resultados restantes.

Si no especifica que se exporten los resultados de la ejecución del flujo de trabajo secundario, la ejecución de la máquina de estado devuelve una matriz de resultados de la ejecución del flujo de trabajo secundario, como se muestra en el siguiente ejemplo:

> **Note**
>
> Si el tamaño de salida devuelto supera los 256 KB, se produce un error en la ejecución de la máquina de estado y se devuelve un error `States.DataLimitExceeded`.

```
[
  {
    "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"
    }
  },
  ...
]
```

Políticas de IAM para ResultWriter

Al crear flujos de trabajo con la consola de Step Functions, Step Functions puede generar automáticamente políticas de IAM basadas en los recursos de la definición de flujo de trabajo. Estas políticas incluyen los privilegios mínimos necesarios para permitir que el rol de la máquina de estado invoque la acción de la API `StartExecution` para el estado Map Distributed. Estas políticas también incluyen los privilegios mínimos necesarios para que Step Functions pueda acceder a los recursos de AWS, como buckets y objetos de Amazon S3 y las funciones de Lambda. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM. Por ejemplo, si el flujo de trabajo incluye un estado `Map` en modo distribuido, aplique las políticas al bucket y a la carpeta de Amazon S3 específicos que contengan el conjunto de datos.

> **Important**
>
> Si especifica un bucket y un objeto de Amazon S3, o un prefijo, con una ruta de referencia a un par clave-valor existente en la entrada del estado Map Distributed, no olvide actualizar las políticas de IAM para el flujo de trabajo. Limite las políticas hasta los nombres de objeto y bucket a los que se dirige la ruta en tiempo de ejecución.

El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios para escribir los resultados de la ejecución del flujo de trabajo secundario en una carpeta denominada *`csvJobs`* de un bucket de Amazon S3 mediante la acción `PutObject` de la API.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Si el bucket de Amazon S3 en el que está escribiendo el resultado de la ejecución del flujo de trabajo secundario está cifrado con una clave AWS Key Management Service (AWS KMS), debe incluir los permisos AWS KMS necesarios en la política de IAM. Para obtener más información, consulte Permisos de IAM para un bucket de Amazon S3 AWS KMS key cifrado.

Análisis de un archivo CSV de entrada

Puesto que no existe un formato estandarizado para crear y mantener los datos en los archivos CSV, Step Functions analiza los archivos CSV según las siguientes reglas:

- Las comas (,) son un delimitador que separa los campos individuales.
- Los retornos de carro son un delimitador que separa los registros individuales.
- Los campos se tratan como cadenas. Para las conversiones de tipos de datos, utilice la función intrínseca `States.StringToJson` en ItemSelector.
- No es necesario incluir comillas dobles (" ") para delimitar cadenas. No obstante, las cadenas entre comillas dobles pueden contener comas y retornos de carro sin que funcionen como delimitadores.
- Para incluir las comillas dobles en una secuencia de escape, repítalas.
- Si el número de campos de una fila es inferior al número de campos del encabezado, Step Functions proporciona cadenas vacías para los valores que faltan.
- Si el número de campos de una fila es mayor que el número de campos del encabezado, Step Functions omite los campos adicionales.

Example de análisis de un archivo CSV de entrada

Supongamos que ha proporcionado un archivo CSV denominado *myCSVInput.csv* que contiene una fila como entrada. A continuación, ha almacenado este archivo en un bucket de Amazon S3 denominado *my-bucket*. El archivo CSV el siguiente.

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

La siguiente máquina de estado lee este archivo CSV y utiliza ItemSelector para convertir los tipos de datos de algunos de los campos.

```
{
  "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)"
      }
    }
  }
}
```

Cuando ejecuta esta máquina de estado, produce el siguiente resultado.

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

Objeto Context (Contexto)

El objeto de contexto es una estructura JSON interna que está disponible durante una ejecución y contiene información sobre la máquina de estados y la ejecución. Esto permite a sus flujos de trabajo

acceder a la información acerca de su ejecución específica. Puede acceder al objeto de contexto desde los siguientes campos:

- `InputPath`
- `OutputPath`
- `ItemsPath` (en los estados Map)
- `Variable` (en los estados Choice)
- `ResultSelector`
- `Parameters`
- Operadores de comparación entre variables

Formato de objetos de contexto

El objeto de contexto incluye información acerca de la máquina de estado, el estado, la ejecución y la tarea. Este objeto JSON incluye nodos para cada tipo de datos y se encuentra en el siguiente formato.

```
{
    "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"
    }
```

```
}
```

Durante una ejecución, el objeto de contexto se rellena con datos relevantes para el campo `Parameters` desde donde se accede. El valor de un campo `Task` es nulo si el campo `Parameters` se encuentra fuera de un estado de tarea.

El valor del objeto de contexto `RedriveCount` es 0 si aún no ha realizado redriven con una ejecución. Además, el objeto de contexto `RedriveTime` solo está disponible si se tiene una ejecución redriven. Si ha utilizado redriven a Map Run, el objeto de contexto `RedriveTime` solo está disponible para los flujos de trabajo secundarios de tipo estándar. Para un redriven Map Run con flujos de trabajo secundarios de tipo Express, `RedriveTime` no está disponible.

El contenido de una ejecución en curso incluye información detallada en el siguiente formato.

```
{
    "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**
>
> Para obtener información sobre los datos de objetos context relacionados con los estados `Map`, consulte Datos del objeto de contexto para los estados Map.

Acceso al objeto de contexto

Para obtener acceso al contexto objeto, especifique en primer lugar el nombre del parámetro añadiendo `.$` al final, de la misma forma que cuando selecciona la entrada del estado con una ruta. A continuación, para obtener acceso a datos de objetos de contexto en lugar de la entrada, anexe la ruta con `$$.`. Esto indica AWS Step Functions que debes usar la ruta para seleccionar un nodo en el objeto de contexto.

Los siguientes ejemplos muestran cómo se puede acceder a los objetos de contexto, como el identificador de ejecución, el nombre, la hora de inicio y el recuento de redrive.

- Recuperación y transferencia del ARN de ejecución a un servicio descendente
- Acceso a la hora y el nombre de inicio de la ejecución en un estado Pass
- Acceso al recuento de redrive de una ejecución

Recuperación y transferencia del ARN de ejecución a un servicio descendente

Este estado de tarea de ejemplo utiliza una ruta para retirar y pasar el nombre de recurso de Amazon (ARN) de ejecución a un mensaje de Amazon Simple Queue Service (Amazon SQS).

```
{
  "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"
```

```
  }
}
```

Para obtener más información acerca de cómo utilizar el token de tarea al llamar a un servicio integrado, consulte Cómo esperar una devolución de llamada con el token de tarea.

Acceso a la hora y el nombre de inicio de la ejecución en un estado Pass

```
{
  "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
    }
  }
}
```

Acceso al recuento de redrive de una ejecución

El siguiente ejemplo de definición de un estado de tarea muestra cómo se puede acceder al recuento de redrive de una ejecución.

```
{
  "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
}
```

Datos del objeto de contexto para los estados Map

Existen dos elementos adicionales disponibles en el objeto context al procesar un estado Map: `Index` y `Value`. Para cada iteración del estado `Map`, `Index` contiene el número de índice del elemento de matriz que se está procesando actualmente, mientras que `Value` contiene el elemento de matriz que se está procesando. Dentro de un estado `Map`, el objeto context incluye los datos siguientes:

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

Solo están disponibles en un estado `Map` y se pueden especificar en el campo `ItemSelector`.

> **Note**
>
> Debe definir parámetros desde el objeto context en el bloque `ItemSelector` del estado `Map` principal, no dentro de los estados incluidos en la sección `ItemProcessor`.

Dada una máquina de estado con un estado `Map` sencillo, podemos inyectar información del objeto context de la siguiente manera.

```
{
  "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
    }
  }
}
```

Si ejecuta la máquina de estado anterior con la siguiente entrada, `Index` y `Value` se insertan en la salida.

```
[
  {
    "who": "bob"
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
]
```

El resultado de la ejecución devuelve los valores de los elementos `Index` y `Value` de cada una de las tres iteraciones de la siguiente manera:

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
      "who": "meg"
    }
  },
```

```
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

Simulador de flujo de datos

Puede diseñar, implementar y depurar flujos de trabajo en la consola de Step Functions. También puede controlar el flujo de datos en sus flujos de trabajo mediante el procesamiento de entrada y salida de JsonPath. Con el simulador de flujo de datos, puede simular el orden en que los estados Estado de la tarea de su flujo de trabajo procesan los datos en tiempo de ejecución. Con el simulador, puede comprender cómo filtrar y manipular los datos a medida que fluyen de un estado a otro. Simula cada uno de los siguientes campos que Step Functions utiliza para procesar y controlar el flujo de datos JSON:

`InputPath`

Selecciona QUÉ parte de toda la carga de entrada se utilizará como entrada de una tarea. Si especifica este campo, Step Functions lo aplicará en primer lugar.

`Parámetros`

Especifica CÓMO debe mostrarse la entrada antes de invocar la tarea. Con el campo `Parameters` puede crear una colección de pares clave-valor que se pasan como entrada a una integración de Servicio de AWS, por ejemplo, a una función AWS Lambda. Estos valores pueden ser estáticos o seleccionarse dinámicamente desde la entrada de estado o desde el objeto de contexto del flujo de trabajo.

`ResultSelector`

Determina QUÉ elegir del resultado de una tarea. Con el campo `ResultSelector` puede crear una colección de pares clave-valor que sustituyan el resultado de un estado y que pasen esa colección a `ResultPath`.

`ResultPath`

Determina DÓNDE colocar el resultado de una tarea. Utilice `ResultPath` para determinar si la salida de un estado es una copia de su entrada, el resultado que produce o una combinación de ambos.

`OutputPath`

Determina QUÉ enviar al siguiente estado. Con `OutputPath` puede filtrar la información no deseada y pasar solo la parte de datos JSON que le interese.

En este tema

Uso del simulador de flujo de datos

El simulador proporciona una comparación lado a lado de los datos en tiempo real antes y después de aplicar un campo de procesamiento de datos de entrada y salida. Para usar el simulador, especifique una entrada JSON. A continuación, evalúelo a través de cada uno de los campos de procesamiento de entrada y salida. El simulador valida automáticamente la entrada de JSON y resalta cualquier error de sintaxis.

Para usar el simulador de flujo de datos

En los siguientes pasos, debe proporcionar una entrada de JSON y aplicar los campos `InputPath` y `Parámetros`. También puede aplicar los demás campos disponibles y ver sus resultados.

1. Abra la consola de Step Functions.
2. En el panel de navegación, elija Simulador de flujo de datos.
3. En el área de Entrada de estado, sustituya los datos JSON de ejemplo rellenados previamente por los siguientes datos JSON. A continuación, haga clic en Siguiente.

```
{
  "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. Para InputPath, **`$.data.address`** introduzca para seleccionar el nodo de dirección de los datos JSON de entrada.

   El cuadro Entrada de estado después de InputPath muestra los siguientes resultados.

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

5. Elija Siguiente.
6. Aplique el campo `Parameters` para convertir los datos JSON resultantes en una cadena. Para convertir los datos, haga lo siguiente:

   - En el cuadro Parámetros, introduzca el siguiente código para crear una cadena llamada `addressString`.

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

7. Vea el resultado de la aplicación del campo `Parameters` en el cuadro Entrada filtrada después de los parámetros.

Consideraciones sobre el uso del simulador de flujo de datos

Antes de utilizar el simulador de flujo de datos, tenga en cuenta sus limitaciones, que incluyen, entre otras, las siguientes:

- Expresiones de filtro no compatibles

  Las expresiones de filtro en el simulador se comportan de forma diferente que en el servicio de Step Functions. Esto incluye expresiones que utilizan la siguiente sintaxis:`[?(expression)]`. A continuación, se muestra una lista de expresiones de base de datos no compatibles. Si se utilizan, es posible que estas expresiones no devuelvan el resultado esperado tras su evaluación.

  - `$..book[?(@.isInStock==true)]`
  - `$..book[?(@.price > 10.0)].title`
- Evaluación de JsonPath incorrecta para matrices de un solo elemento

  Si filtra los datos con una expresión JsonPath que devuelva un único elemento de matriz, el simulador devolverá el elemento sin la matriz. Por ejemplo, fíjese en la siguiente matriz de datos, denominada `items`:

```
{
  "items": [
    {
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

  Con esta matriz `items`, si ingresa **`$..comment`** en el campo InputPath, esperaría obtener el siguiente resultado:

```
[
  "nice shoe"
]
```

Sin embargo, el simulador de flujo de datos devuelve el siguiente resultado en su lugar:

```
"nice shoe"
```

Para la evaluación por JsonPath de una matriz que contiene varios elementos, el simulador devuelve el resultado esperado.

Gestión de implementaciones continuas con versiones y alias

Puede utilizar Step Functions para gestionar las implementaciones continuas de los flujos de trabajo mediante versiones y alias de máquina de estado. Una versión es una instantánea numerada e inmutable de una máquina de estado que se puede ejecutar. Un alias es un puntero para un máximo de dos versiones de una máquina de estado.

Puede mantener varias versiones de las máquinas de estado y gestionar su implementación en el flujo de trabajo de producción. Con alias, puede enrutar tráfico entre diferentes versiones del flujo de trabajo e implementar gradualmente esos flujos de trabajo en el entorno de producción.

Además, puede iniciar ejecuciones de máquinas de estado mediante una versión o un alias. Si no se utiliza una versión o un alias al iniciar la ejecución de una máquina de estado, Step Functions utiliza la última revisión de la definición de la máquina de estado.

> **Revisión de máquina de estado**
>
> Una máquina de estado puede tener una o más revisiones. Al actualizar una máquina de estado mediante la acción de la API UpdateStateMachine, se crea una nueva revisión de la máquina de estado. Una revisión es una instantánea inmutable y de solo lectura de la definición y configuración de una máquina de estado. No se puede iniciar la ejecución de una máquina de estado a partir de una revisión y las revisiones no tienen un ARN. Las revisiones tienen un `revisionId`, que es un identificador único universal (UUID).

Contenido

- Versiones de máquinas de estado
- Alias de máquinas de estado
- Autorización para versiones y alias
- Asociar ejecuciones de máquinas de estado a una versión o alias
- Ejemplo de implementación de alias y versiones
- Realiza una implementación gradual de las versiones de las máquinas de estado

Versiones de máquinas de estado

Una versión es una instantánea numerada e inmutable de una máquina de estado. Usted publica versiones de la revisión más reciente realizada en esa máquina de estado. Cada versión tiene su propio nombre de recurso de Amazon (ARN). Este ARN es una combinación del ARN de la máquina de estado y el número de versión separados por dos puntos (:). En el ejemplo siguiente se muestra el formato de un ARN de versión de máquina de estado.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

Para empezar a utilizar versiones de máquinas de estado, debe publicar la primera versión. Tras publicar una versión, puede invocar la acción de la StartExecutionAPI con el ARN de la versión. No se puede editar una versión, pero es posible actualizar una máquina de estado y publicar una versión nueva. También se puede publicar varias versiones de la máquina de estado.

Cuando se publica una nueva versión de la máquina de estado, Step Functions le asigna un número de versión. Los números de versión comienzan en 1 y aumentan de forma monótona para cada nueva versión. Los números de versión no se reutilizan para una máquina de estados dada. Si se elimina la versión 10 de la máquina de estado y, a continuación, se publica una nueva versión, Step Functions la publica como versión 11.

Las siguientes propiedades son las mismas para todas las versiones de una máquina de estado:

- Todas las versiones de una máquina de estado comparten el mismo tipo (estándar o rápido).
- No se puede cambiar el nombre o la fecha de creación de una máquina de estado de una versión a otra.
- Las etiquetas se aplican globalmente a las máquinas de estado. Puede administrar las etiquetas de las máquinas de estado mediante las acciones TagResourcey de la UntagResourceAPI.

Las máquinas de estado también contienen propiedades que forman parte de cada versión y revision, pero estas propiedades pueden diferir entre dos versiones o revisiones determinadas. Estas propiedades incluyen la Definición de la máquina de estado, el rol de IAM, la configuración de seguimiento y la configuración de registro.

Contenido

Publicación de una versión de una máquina de estado (Consola)

Puede publicar hasta 1000 versiones de una máquina de estado. Para solicitar un aumento de este límite flexible, utilice la página Centro de soporte de AWS Management Console. Puedes eliminar manualmente las versiones no utilizadas de la consola o invocando la acción de la DeleteStateMachineVersionAPI.

Para publicar una versión de una máquina de estado

1. Abra la consola de Step Functions y, a continuación, elija una máquina de estado existente.
2. En la página Detalle de la máquina de estado, elija Editar.
3. Edite la definición de la máquina de estados según sea necesario y, a continuación, elija Guardar.
4. Elija Publicar versión.
5. (Opcional) En el campo Descripción del cuadro de diálogo que aparece, escriba una breve descripción de la versión de la máquina de estado.
6. Elija Publicar.

> **Note**
>
> Cuando se publica una nueva versión de la máquina de estado, Step Functions le asigna un número de versión. Los números de versión comienzan en 1 y aumentan de forma monótona para cada nueva versión. Los números de versión no se reutilizan para una máquina de estados dada. Si se elimina la versión 10 de la máquina de estado y, a continuación, se publica una nueva versión, Step Functions la publica como versión 11.

Gestión de versiones con operaciones de API de Step Functions

Step Functions proporciona las siguientes operaciones de API para publicar y gestionar versiones de máquinas de estado:

- PublishStateMachineVersion— Publica una versión a partir de la versión actual revision de una máquina de estado.
- UpdateStateMachine— Publica una nueva versión de la máquina de estados si se actualiza una máquina de estados y se establece el `publish` parámetro `true` en la misma solicitud.
- CreateStateMachine— Publica la primera revisión de la máquina de estados si se establece el `publish` parámetro en`true`.
- ListStateMachineVersions— Muestra las versiones del ARN de la máquina de estado especificada.
- DescribeStateMachine— Devuelve los detalles de la versión de la máquina de estados para una versión ARN especificada en. `stateMachineArn`
- DeleteStateMachineVersion— Elimina una versión de máquina de estados.

Para publicar una nueva versión de la revisión actual de una máquina de estados llamada *`myStateMachine`* mediante el AWS Command Line Interface, utilice el `publish-state-machine-version` comando:

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

La respuesta devuelve el `stateMachineVersionArn`. Por ejemplo, el comando anterior devuelve una respuesta de`arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

> **Note**
>
> Cuando se publica una nueva versión de la máquina de estado, Step Functions le asigna un número de versión. Los números de versión comienzan en 1 y aumentan de forma monótona para cada nueva versión. Los números de versión no se reutilizan para una máquina de estados dada. Si se elimina la versión 10 de la máquina de estado y, a continuación, se publica una nueva versión, Step Functions la publica como versión 11.

Ejecución de una versión de una máquina de estado desde la consola

Para empezar a utilizar versiones de máquina de estado, primero debe publicar una versión de la máquina de estado actual revision. Para publicar una versión, utilice la consola Step Functions o invoque la acción de la PublishStateMachineVersionAPI. También puede invocar la acción de la UpdateStateMachineAliasAPI con un parámetro opcional denominado `publish` para actualizar una máquina de estados y publicar su versión.

Puede iniciar las ejecuciones de una versión mediante la consola o invocando la acción de la StartExecutionAPI y proporcionando el ARN de la versión. También puede usar un alias para iniciar las ejecuciones de una versión. Según su configuración de direccionamiento, un alias dirige el tráfico a una versión específica.

Si inicia la ejecución de una máquina de estado sin usar una versión, Step Functions utilizará la revisión más reciente de la máquina de estado para la ejecución. Para obtener información sobre cómo Step Functions asocia una ejecución a una versión, consulte Asociar ejecuciones a una versión o alias.

Para iniciar una ejecución utilizando una versión de una máquina de estado

1. Abra la consola de Step Functions y, a continuación, elija una máquina de estado existente de la que haya publicado una o más versiones. Para aprender a publicar una versión, consulte Publicación de una versión de una máquina de estado (Consola).
2. En la página Detalle de la máquina de estado, elija la pestaña Versiones.
3. En la sección Versiones, haga lo siguiente:
   a. Seleccione la versión con la que desee iniciar la ejecución.
   b. Seleccione Iniciar ejecución.
4. (Opcional) En el cuadro de diálogo Iniciar ejecución, escriba un nombre para la ejecución.

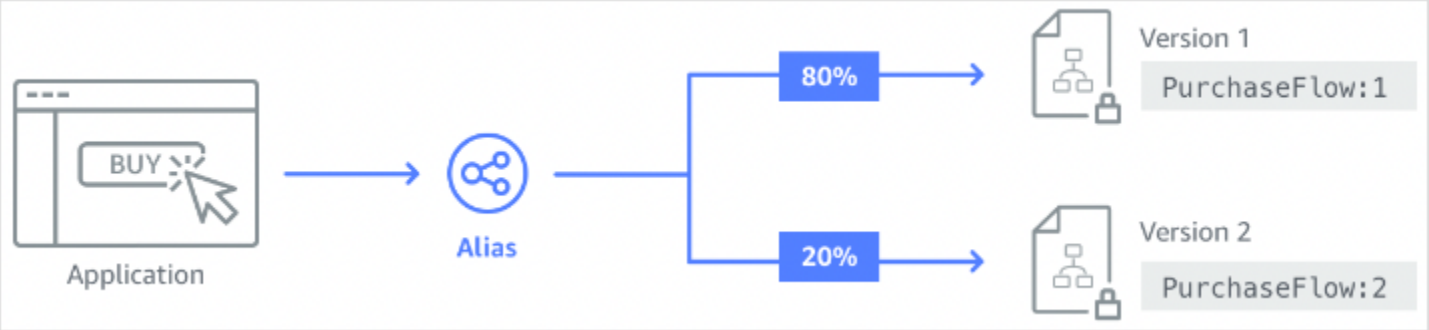
5. (Opcional) Introduzca la entrada de ejecución y, a continuación, seleccione Iniciar ejecución.

Alias de máquinas de estado

Un alias es un puntero para un máximo de dos versiones de la misma máquina de estado. Puede crear varios alias para las máquinas de estado. Cada alias tiene un nombre de recurso de Amazon (ARN) único. El ARN de alias es una combinación del ARN de la máquina de estado y el nombre del alias, separados por dos puntos (:). En el ejemplo siguiente se muestra el formato de un alias de máquina de estado.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

Puede usar un alias para dirigir tráfico entre una de las dos versiones de la máquina de estado. También puede crear un alias que apunte a una única versión. Los alias solo pueden apuntar a versiones de máquinas de estado. No se puede usar un alias para apuntar a otro alias. También se puede actualizar un alias para que apunte a una nueva versión de la máquina de estado.



Contenido

- Creación de un alias de máquina de estado (Consola)
- Gestión de alias con las operaciones de la API de Step Functions
- Configuración de direccionamiento de alias
- Ejecución de una máquina de estado con un alias (Consola)

Creación de un alias de máquina de estado (Consola)

Puede crear hasta 100 alias para cada máquina de estado mediante la consola Step Functions o invocando la acción de la CreateStateMachineAliasAPI. Para solicitar un aumento de este límite flexible, utilice la página Centro de soporte de AWS Management Console. Elimine los alias no utilizados de la consola o invoque la acción de la API. DeleteStateMachineAlias

Para crear un alias de máquina de estado

1. Abra la consola de Step Functions y, a continuación, elija una máquina de estado existente.
2. En la página Detalle de la máquina de estado, elija la pestaña Alias.
3. Elija Crear nuevo alias.
4. En la página Crear alias, haga lo siguiente:
   a. Escriba un Nombre de alias.
   b. (Opcional) En Descripción, escriba una descripción del alias.
5. Para configurar el direccionamiento en el alias, consulte Configuración de direccionamiento de alias.
6. Elija Crear alias.

Gestión de alias con las operaciones de la API de Step Functions

Step Functions proporciona las siguientes operaciones de API que puede utilizar para crear y gestionar alias de máquinas de estado u obtener información sobre los alias:

- CreateStateMachineAlias— Crea un alias para una máquina de estados.
- DescribeStateMachineAlias— Devuelve detalles sobre el alias de una máquina de estados.
- ListStateMachineAliases— Muestra los alias del ARN de la máquina de estado especificada.
- UpdateStateMachineAlias— Actualiza la configuración de un alias de máquina de estado existente modificando su `description` o. `routingConfiguration`
- DeleteStateMachineAlias— Elimina una versión de la máquina de estados.

Para crear un alias con un nombre *`PROD`* que apunte a la versión 1 de una máquina de estados denominada *`myStateMachine`* mediante el AWS Command Line Interface, utilice el `create-state-machine-alias` comando.

```
aws stepfunctions create-state-machine-alias --name PROD  --routing-
configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

Configuración de direccionamiento de alias

Puede usar un alias para dirigir tráfico de ejecución entre dos versiones de una máquina de estado. Por ejemplo, supongamos que desea lanzar una nueva versión de la máquina de estado. Puede reducir los riesgos que implica la implementación de la nueva versión configurando el direccionamiento en un alias. Al configurar el direccionamiento puede enviar la mayor parte del tráfico a una versión anterior y probada de la máquina de estado. De este modo, la nueva versión puede recibir un porcentaje menor, hasta que se confirme que es seguro lanzar la nueva versión.

Para definir la configuración de direccionamiento, publique las dos versiones de la máquina de estado a las que apunta el alias. Al iniciar una ejecución desde un alias, Step Functions elige aleatoriamente la versión de la máquina de estado que se va a ejecutar a partir de las versiones especificadas en la configuración de direccionamiento. Basa esta elección en el porcentaje de tráfico que se asigna a cada versión en la configuración de direccionamiento de alias.

Para configurar la configuración de direccionamiento en un alias

- En la página Crear alias, en Configuración de direccionamiento, haga lo siguiente:

  a. En Versión, elija la primera versión de la máquina de estado a la que apunte el alias.

  b. Active la casilla de verificación Dividir el tráfico entre dos versiones.

  > **Tip**
  >
  > Para seleccionar una sola versión, desactive la casilla de verificación Dividir el tráfico entre dos versiones.

  c. En Versión, elija la segunda versión a la que debe apuntar el alias.

  d. En los campos de Porcentaje de tráfico, especifique el porcentaje de tráfico que se direccionará a cada versión. Por ejemplo, introduzca **`60`** y **`40`** para direccionar el 60 por ciento del tráfico de ejecución a la primera versión y el 40 por ciento del tráfico a la segunda versión.

     Los porcentajes de tráfico combinados deben ser iguales al 100 por ciento.

Ejecución de una máquina de estado con un alias (Consola)

Puede iniciar las ejecuciones de máquinas de estado con un alias desde la consola o invocando la acción de la StartExecutionAPI con el ARN del alias. A continuación, Step Functions ejecuta la versión especificada por el alias. De forma predeterminada, si no especifica una versión o un alias al iniciar la ejecución de una máquina de estado, Step Functions utilizará la revisión más reciente.

Para iniciar la ejecución de una máquina de estado utilizando un alias

1. Abra la consola de Step Functions y, a continuación, seleccione una máquina de estado existente para la que haya creado un alias. Para obtener información acerca de la creación de un alias, consulte Creación de un alias de máquina de estado (Consola).
2. En la página Detalle de la máquina de estado, elija la pestaña Alias.
3. En la sección Alias, haga lo siguiente:
   a. Seleccione el alias con el que desee comenzar la ejecución.
   b. Seleccione Iniciar ejecución.
4. (Opcional) En el cuadro de diálogo Iniciar ejecución, escriba un nombre para la ejecución.
5. Si es necesario, introduzca la entrada de ejecución y, a continuación, seleccione Iniciar ejecución.

Autorización para versiones y alias

Para invocar las acciones de la API de Step Functions con una versión o un alias, necesita los permisos adecuados. Para autorizar una versión o un alias a invocar una acción de API, Step Functions usa el ARN de la máquina de estado en lugar de usar el ARN de la versión o el ARN del alias. También puede restringir los permisos para una versión o un alias específicos. Para obtener más información, consulte Reducir el alcance de los permisos.

Puede utilizar el siguiente ejemplo de política de IAM de una máquina de estado denominada *myStateMachine* para invocar la acción de la API CreateStateMachineAlias con el propósito de crear un alias de máquina de estado.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "states:CreateStateMachineAlias",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine"
    }
  ]
}
```

Cuando configure permisos para permitir o denegar el acceso a acciones de la API mediante versiones o alias de máquinas de estado, tenga en cuenta lo siguiente:

- Si usa el parámetro `publish` de las acciones de la API CreateStateMachine y UpdateMachine para publicar una nueva versión de la máquina de estado, también necesitará el permiso `ALLOW` de la acción de la API PublishStateMachineVersion.
- La acción de la API DeleteStateMachine elimina todas las versiones y los alias asociados a una máquina de estado.

En este tema

Reducir el alcance de los permisos de una versión o un alias

Puede usar un calificador para reducir más el permiso de autorización que necesita una versión o un alias. Un calificador hace referencia a un número de versión o a un nombre de alias. El calificador se utiliza para calificar una máquina de estado. El siguiente ejemplo es un ARN de máquina de estado que usa un alias denominado `PROD` como calificador.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

Para obtener más información sobre ARN calificados y no calificados, consulte Asociar ejecuciones a una versión o alias.

Para reducir el alcance de los permisos, utilice la clave de contexto opcional llamada `states:StateMachineQualifier` en la instrucción `Condition` de una política de IAM. Por ejemplo, la siguiente política de IAM para una máquina de estado denominada `myStateMachine` deniega el acceso para invocar la acción de la API DescribeStateMachine con un alias llamado `PROD` o la versión 1.

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

En la siguiente lista se especifican las acciones de la API que se pueden utilizar para limitar los permisos con la clave de contexto `StateMachineQualifier`.

- CreateStateMachineAlias
- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

Asociar ejecuciones de máquinas de estado a una versión o alias

Step Functions asocia una ejecución a una versión o alias en función del nombre de recurso de Amazon (ARN) que se utiliza para invocar la acción de la StartExecutionAPI. Step Functions realiza esta acción en el momento en que comienza la ejecución.

Puede iniciar la ejecución de una máquina de estado mediante un ARN completo o incompleto.

- ARN completo: hace referencia al ARN de una máquina de estado que tiene sufijo con un número de versión o un nombre de alias.

  El siguiente ejemplo de ARN completo hace referencia a la versión 3 de una máquina de estado denominada `myStateMachine`.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
  ```

  El siguiente ejemplo de ARN completo hace referencia a un alias denominado `PROD` de una máquina de estado llamada `myStateMachine`.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
  ```

- ARN incompleto: hace referencia al ARN de una máquina de estado sin número de versión ni sufijo de nombre de alias.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
  ```

Por ejemplo, si su ARN incompleto hace referencia a la versión 3, Step Functions asocia la ejecución a esta versión. No asocia la ejecución a ningún alias que apunte a la versión 3.

Si su ARN completo hace referencia a un alias, Step Functions asocia la ejecución a ese alias y a la versión a la que apunta el alias. Una ejecución solo puede asociarse a un alias.

> **Note**
>
> Si inicia una ejecución con un ARN incompleto, Step Functions no asocia esa ejecución a una versión aunque la versión utilice la misma revision de la máquina de estado. Por ejemplo, si la versión 3 utiliza la última revisión, pero inicia una ejecución con un ARN incompleto, Step Functions no asocia esa ejecución a la versión 3.

En este tema

Ver las ejecuciones iniciadas con una versión o un alias

Step Functions ofrece las siguientes formas de ver las ejecuciones iniciadas con una versión o un alias:

Uso de acciones de la API

Puede ver todas las ejecuciones asociadas a una versión o un alias invocando las acciones de la API DescribeExecutiony ListExecutions. Estas acciones de la API devuelven el ARN de la versión o el alias que se utilizó para iniciar la ejecución. Estas acciones también devuelven otros detalles, como el estado y el ARN de la ejecución.

También puede proporcionar un ARN de alias de máquina de estado o un ARN de versión para enumerar las ejecuciones asociadas a un alias o una versión específicos.

El siguiente ejemplo de respuesta de la acción de la ListExecutionsAPI muestra el ARN del alias utilizado para iniciar una ejecución de máquina de estado denominada. *myFirstExecution*

El texto en *cursiva* del siguiente fragmento de código representa información específica del recurso.

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

Uso de la consola de Step Functions

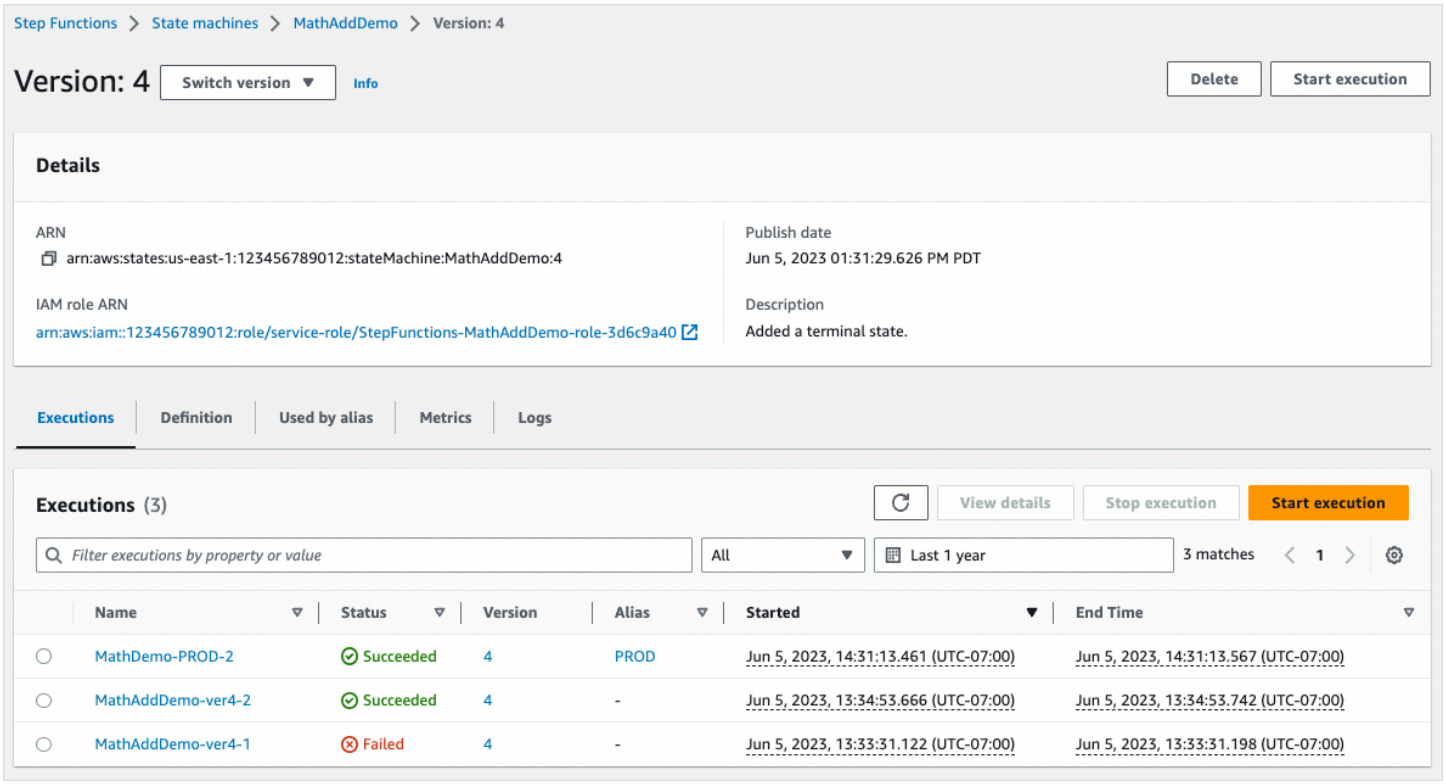
También puede ver las ejecuciones iniciadas por una versión o un alias desde la consola de Step Functions. En el siguiente procedimiento se muestra cómo ver las ejecuciones iniciadas con una versión específica:

1. Abra la consola de Step Functions y, a continuación, seleccione una máquina de estado existente para la que haya publicado una versión o creado un alias. En este ejemplo se muestra cómo ver las ejecuciones iniciadas con una versión específica de la máquina de estado.
2. Seleccione la pestaña Versiones y, a continuación, elija una versión de la lista de versiones.

   > **Tip**
   >
   > Filtre por propiedad o cuadro de valores para buscar una versión específica.

3. En la página Detalles de la versión, puede ver una lista de todas las ejecuciones de máquinas de estado en curso y pasadas que se hayan iniciado con la versión seleccionada.

En la siguiente imagen se muestra la página Detalles de la versión de la consola. Esta página muestra las ejecuciones iniciadas por la versión 4 de una máquina de estado denominada *`MathAddDemo`*. Esta lista también muestra una ejecución que se inició con un alias denominado *`PROD`*. Este alias enrutaba el tráfico de ejecución a la versión 4.



Uso de las métricas de CloudWatch

Para cada ejecución de una máquina de estado que se inicie con un Qualified ARN, Step Functions emite métricas adicionales con el mismo nombre y valor que las métricas emitidas actualmente. Estas métricas adicionales contienen dimensiones para cada identificador de la versión y nombre de alias con los que se inicia una ejecución. Con estas métricas, puede supervisar las ejecuciones de la máquinas de estado en el nivel de versión y determinar cuándo podría ser necesaria una reversión. También puedes crear CloudWatch alarmas de Amazon en función de estas métricas.

Step Functions emite las siguientes métricas para las ejecuciones que se inician con un alias o una versión:

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`
- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

Si ha iniciado la ejecución con un ARN de versión, Step Functions publica la métrica con `StateMachineArn` y una segunda métrica con las dimensiones `StateMachineArn` y`Version`.

Si ha iniciado la ejecución con un ARN de alias, Step Functions emite las siguientes métricas:

- Dos métricas para el ARN incompleto y la versión.
- Una métrica con las dimensiones `StateMachineArn` y `Alias`.

Ejemplo de implementación de alias y versiones

En el siguiente ejemplo de la técnica de implementación de valores controlados se muestra cómo se puede implementar una nueva versión de la máquina de estado con la AWS Command Line Interface. En este ejemplo, el alias que cree enruta el 20 por ciento del tráfico de ejecución a la nueva versión. A continuación, enruta el 80 por ciento restante a la versión anterior. Para implementar una nueva versión de la máquina de estados y cambiar el tráfico de ejecución a un alias, complete los siguientes pasos:

1. Publique una versión de la revisión actual de la máquina de estado.

   Utilice el comando publish-state-machine-version en la AWS CLI para publicar una versión de la revisión actual de una máquina de estado llamada *myStateMachine*:

   ```
   aws stepfunctions publish-state-machine-version --state-machine-arn
    arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
   ```

   La respuesta devuelve `stateMachineVersionArn` de la versión que ha publicado. Por ejemplo, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

2. Cree un alias que apunte a la nueva versión de la máquina de estado.

   Utilice el comando create-state-machine-alias para crear un nombre de alias *PROD* que apunte a la versión 1 de *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. Compruebe que las ejecuciones iniciadas con el alias utilizan la versión publicada correcta.

Inicie una nueva ejecución de *myStateMachine* proporcionando el ARN del alias **PROD** en el comando start-execution:

```
aws stepfunctions start-execution
  --state-machine-arn arn:aws:states:us-
east-1:123456789012:stateMachineAlias:myStateMachine:PROD
  --input "{}"
```

Si proporciona el ARN de la máquina de estado en la solicitud StartExecution, utiliza la revision más reciente de la máquina de estado en lugar de la versión especificada en su alias para iniciar la ejecución.

4. Actualice la definición de la máquina de estado y publique una nueva versión.

   Actualice *myStateMachine* y publique su nueva versión. Para ello, utilice el parámetro `publish` opcional del comando update-state-machine:

   ```
   aws stepfunctions update-state-machine
       --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine
       --definition $UPDATED_STATE_MACHINE_DEFINITION
       --publish
   ```

   La respuesta devuelve `stateMachineVersionArn` para la nueva versión. Por ejemplo, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:2`.

5. Actualice el alias para que apunte a ambas versiones y establezca la configuración de enrutamiento del alias.

   Utilice el comando update-state-machine-alias para actualizar la configuración de enrutamiento del alias PROD. Configure el alias para que el 80 por ciento del tráfico de ejecución vaya a la versión 1 y el 20 por ciento restante a la versión 2:

   ```
   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. Reemplace la versión 1 por la versión 2.

Tras comprobar que la nueva versión de la máquina de estado funciona correctamente, puede implementar la nueva versión de esta. Para ello, vuelva a actualizar el alias para asignar el 100 por ciento del tráfico de ejecución a la nueva versión.

Use el comando update-state-machine-alias para establecer la configuración de enrutamiento del alias PROD en el 100 por ciento para la versión 2:

```
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

Para revertir la implementación de la versión 2, edite la configuración de enrutamiento del alias para transferir el 100 por ciento del tráfico a la versión recién implementada.

```
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}]"
```

Puede usar versiones y alias para realizar otros tipos de implementaciones. Por ejemplo, puede realizar una implementación continua de una nueva versión de su máquina de estado. Para ello, aumente gradualmente el porcentaje ponderado en la configuración de enrutamiento del alias que apunta a la nueva versión.

También puede usar versiones y alias para realizar una implementación azul/verde. Para ello, cree un alias denominado `green` que ejecute la versión 1 actual de su máquina de estado. A continuación, cree otro alias denominado `blue` que ejecute la nueva versión, por ejemplo, *2*. Para probar la nueva versión, envíe el tráfico de ejecución al alias `blue`. Cuando esté seguro de que la nueva versión funciona correctamente, actualice el alias `green` para que apunte a la nueva versión.

Realiza una implementación gradual de las versiones de las máquinas de estado

Una implementación continua es una estrategia de implementación que reemplaza lentamente las versiones anteriores de una aplicación por versiones nuevas de la misma. Para realizar una implementación continua de una versión de máquina de estado, envíe gradualmente una cantidad creciente de tráfico de ejecución a la nueva versión. La cantidad de tráfico y la tasa de aumento son parámetros que configura.

Puede realizar la implementación continua de una versión si opta por una de las opciones siguientes:

- Consola de Step Functions: cree un alias que apunte a dos versiones de la misma máquina de estado. Para este alias, debe configurar la configuración de enrutamiento para transferir el tráfico entre las dos versiones. Para más información acerca de cómo utilizar la consola para desplegar versiones, consulte Versiones y Alias.
- Scripts para AWS CLI y SDK: cree un script de intérprete de comandos con el AWS CLI o el SDK de AWS. Para obtener más información, consulte las siguientes secciones para utilizar AWS CLI y el SDK de AWS.
- Plantillas de AWS CloudFormation: utilice los recursos `AWS::StepFunctions::StateMachineVersion` y `AWS::StepFunctions::StateMachineAlias` para publicar varias versiones de máquinas de estado y crear un alias que apunte a una o dos de estas versiones.

Utilizar la AWS CLI para implementar una nueva versión de la máquina de estado

El script de ejemplo de esta sección muestra cómo se puede utilizar la AWS CLI para cambiar gradualmente el tráfico de una versión anterior de una máquina de estado a una nueva versión de máquina de estado. Puede utilizar este script de ejemplo o actualizarlo según sus necesidades.

Este script muestra una implementación de valores controlados para implementar una nueva versión de máquina de estado mediante un alias. En los siguientes pasos se describen las tareas que realiza el script:

1. Si el parámetro `publish_revision` está establecido en true, publique la versión más reciente revision como la siguiente de la máquina de estado. Si la implementación se realiza correctamente, esta versión se convierte en la nueva versión activa.

Si establece el parámetro `publish_revision` en false, el script implementa la última versión publicada de la máquina de estado.

2. Cree un alias si todavía no existe. Si el alias no existe, dirija el 100 por ciento del tráfico de este alias a la nueva versión y, a continuación, salga del script.
3. Actualice la configuración de enrutamiento del alias para cambiar un pequeño porcentaje del tráfico de la versión anterior a la nueva. Este porcentaje de valor controlado se establece con el parámetro `canary_percentage`.
4. De forma predeterminada, supervise las alarmas configurables de CloudWatch cada 60 segundos. Si se activa alguna de estas alarmas, revierta la implementación inmediatamente dirigiendo el 100 por ciento del tráfico a la versión anterior.

   Después de cada intervalo de tiempo, en segundos, definido en `alarm_polling_interval`, continúe supervisando las alarmas. Continúe la supervisión hasta que haya transcurrido el intervalo de tiempo definido en `canary_interval_seconds`.
5. Si no se activó ninguna alarma durante el proceso `canary_interval_seconds`, transfiera el 100 por ciento del tráfico a la nueva versión.
6. Si la nueva versión se implementa correctamente, elimine las versiones anteriores al número especificado en el parámetro `history_max`.

```
#!/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!
```

Utilizar el SDK de AWS para implementar una nueva versión de la máquina de estado

El script de ejemplo que se encuentra en aws-stepfunctions-examples muestra cómo usar el SDK de AWS para Python para cambiar gradualmente el tráfico de una versión anterior a una nueva versión de una máquina de estado. Puede utilizar este script de ejemplo o actualizarlo según sus necesidades.

El script muestra las siguientes estrategias de implementación:

- Valores controlados: desvía el tráfico en dos incrementos.

  En el primer incremento, un pequeño porcentaje del tráfico, por ejemplo, el 10 por ciento, se desvía a la nueva versión. En el segundo incremento, antes de que supere un intervalo de tiempo específico en segundos, el tráfico restante se desvía a la nueva versión. El cambio a la nueva versión para el tráfico restante solo se produce si no se activa ninguna alarma de CloudWatch durante el intervalo de tiempo especificado.

- Lineal o continuo: desplaza el tráfico a la nueva versión en incrementos iguales, con el mismo número de segundos entre cada incremento.

  Por ejemplo, si especifica el porcentaje de incremento como **`20`** con un `--interval` de **`600`** segundos, esta implementación aumenta el tráfico un 20 por ciento cada 600 segundos hasta que la nueva versión reciba el 100 por ciento del tráfico.

  Esta implementación revierte inmediatamente la nueva versión si se activa alguna alarma de CloudWatch.
- Todo a la vez o azul/verde: transfiere el 100 por ciento del tráfico a la nueva versión de forma inmediata. Esta implementación supervisa la nueva versión y la revierte automáticamente a la versión anterior si se activa alguna alarma de CloudWatch.

Utilizar AWS CloudFormation para implementar una nueva versión de la máquina de estado

En el siguiente ejemplo de plantilla de CloudFormation se publican dos versiones de una máquina de estado denominada *`MyStateMachine`*. Crea un alias denominado *`PROD`*, que apunta a ambas versiones, y luego implementa la versión 2.

En este ejemplo, el 10 por ciento del tráfico se transfiere a la versión 2 cada cinco minutos hasta que esta versión recibe el 100 por ciento del tráfico. Este ejemplo muestra también cómo puede configurar las alarmas de CloudWatch. Si alguna de las alarmas que configuró pasa a ese estado `ALARM`, la implementación fallará y se revertirá inmediatamente.

```
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
```

Ejecuciones en Step Functions

Una ejecución de una máquina de estado se produce cuando se ejecuta una máquina de estado de AWS Step Functions y se realizan sus tareas. Cada máquina de estado de Step Functions puede tener varias ejecuciones simultáneas que se pueden iniciar desde la consola de Step Functions o mediante los SDK de AWS, las acciones de la API de Step Functions o AWS Command Line Interface (AWS CLI). Una ejecución recibe una entrada JSON y produce una salida JSON. Puede iniciar una ejecución de Step Functions de las siguientes maneras:

- Llame a la acción de la API StartExecution.
- Inicie una nueva ejecución en la consola de Step Functions.
- Utilice Amazon EventBridge para iniciar una ejecución en respuesta a un evento.

- Utilice Amazon EventBridge Scheduler para iniciar la ejecución de máquina de estado de acuerdo con un programa.
- Inicie una ejecución con Amazon API Gateway.
- Inicie una ejecución de flujo de trabajo anidado desde un estado Task.

Para obtener más información sobre los diferentes modos de trabajar con Step Functions, consulte Opciones de desarrollo.

Iniciar ejecuciones de flujo de trabajo desde un estado de tarea

AWS Step Functions puede iniciar ejecuciones de flujo de trabajo directamente desde el estado `Task` de una máquina de estado. Esto le permite dividir los flujos de trabajo en máquinas de estado más pequeñas e iniciar ejecuciones de estas otras máquinas de estado. Al iniciar estas nuevas ejecuciones de flujo de trabajo puede:

- Separar el flujo de trabajo de nivel superior de los flujos de trabajo específicos de tareas de nivel inferior.
- Evitar elementos repetitivos llamando a una máquina de estado independiente varias veces.
- Crear una biblioteca de flujos de trabajo reutilizables modulares para un desarrollo más rápido.
- Reducir la complejidad y facilitar la edición y la resolución de problemas de máquinas de estado.

Step Functions puede iniciar estas ejecuciones de flujo de trabajo llamando a su propia API como un servicio integrado. Solo tiene que llamar a la acción de la API `StartExecution` desde su estado `Task` y transferir los parámetros necesarios. Puede llamar a la API de Step Functions utilizando cualquiera de los patrones de integración de servicios.

> **Tip**
>
> Para implementar un ejemplo de un flujo de trabajo anidadoCuenta de AWS, consulte el Módulo 13: Flujos de trabajo rápidos anidados.

Para iniciar una nueva ejecución de una máquina de estado, utilice un estado `Task` similar al ejemplo siguiente.

```
{
```

```
   "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
}
```

Este estado `Task` comenzará una nueva ejecución de la máquina de estado `HelloWorld` y transferirá el comentario JSON como entrada.

> **Note**
>
> Las cuotas de acción de la API `StartExecution` pueden limitar el número de ejecuciones que puede iniciar. Utilice `Retry` en `StepFunctions.ExecutionLimitExceeded` para asegurarse de que se inicia la ejecución. Consulte lo siguiente.
>
> - Cuotas relacionadas con la limitación controlada de las acciones de la API
> - Control de errores en Step Functions

Asociar ejecuciones de flujos de trabajo

Para asociar una ejecución de flujo de trabajo iniciada con la ejecución que la inició, transfiera el ID de ejecución del objeto context a la entrada de ejecución. Puede acceder al ID desde el objeto context desde su estado `Task` en una ejecución en curso. Transfiera el ID de ejecución añadiendo `.$` al nombre del parámetro y haciendo referencia al ID en el objeto context con `$$.Execution.Id`.

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

Puede utilizar un parámetro especial denominado `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID` al iniciar una ejecución. Si se incluye, esta asociación proporciona enlaces en la sección Detalles del paso de la consola de Step Functions. Cuando se proporciona, puede rastrear fácilmente las ejecuciones de los flujos de trabajo desde las ejecuciones de inicio a las ejecuciones de flujo de trabajo iniciadas. Utilizando el ejemplo anterior, asocie el ID de ejecución con la ejecución iniciada de la máquina de estado `HelloWorld` de la siguiente manera.

```
{
   "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
}
```

Para obtener más información, consulte los siguientes temas:

- Trabajo con otros servicios
- Cómo pasar parámetros a una API de servicio
- Acceso al objeto de contexto
- AWS Step Functions

Uso de Programador de Amazon EventBridge con AWS Step Functions

El Programador de Amazon EventBridge es un programador sin servidor que le permite crear, ejecutar y administrar tareas desde un servicio administrado y centralizado. Con el Programador de EventBridge, puede crear programadores mediante expresiones cron y rate para patrones recurrentes, o configurar invocaciones únicas. Puede configurar intervalos de tiempo flexibles para la entrega, definir límites de reintentos y establecer el tiempo máximo de retención para las invocaciones de la API.

Por ejemplo, con Programador de EventBridge, puede iniciar la ejecución de una máquina de estado según una programación cuando se produzca un evento relacionado con la seguridad o para automatizar un trabajo de procesamiento de datos.

En esta página, se explica cómo utilizar el Programador de EventBridge para comenzar la ejecución de una máquina de estado de Step Functions según una programación.

Temas

Configurar el rol de ejecución

Al crear una programación nueva, el Programador de EventBridge debe tener permiso para invocar la operación de la API de destino en su nombre. Estos permisos se conceden al Programador de EventBridge mediante un rol de ejecución. La política de permisos que adjunta a la función de ejecución de su programación define los permisos necesarios. Estos permisos dependen de la API de destino que quiera que invoque el Programador de EventBridge.

Al utilizar la consola del Programador de EventBridge para crear una programación, como en el siguiente procedimiento, el Programador de EventBridge configura de forma automática un rol de ejecución en función del destino seleccionado. Si desea crear una programación con uno de los SDK del Programador de EventBridge, la AWS CLI o AWS CloudFormation, debe tener un rol de ejecución existente que conceda los permisos que el Programador de EventBridge requiere para invocar un destino. A fin de obtener más información sobre cómo configurar de forma manual un rol de ejecución para su programación, consulte Setting up the execution role en EventBridge Scheduler User Guide.

Crear una programación

Para crear una programación con la consola, realice lo siguiente:

1. Abra la consola del Programador de Amazon EventBridge enhttps://console.aws.amazon.com/scheduler/home.
2. En la página de Programaciones, elija Crear programación.

3. En la página de Especificar los detalles de la programación, en la sección de Nombre y descripción de la programación, realice lo siguiente:

   a. En Nombre de la programación, escriba un nombre para la programación. Por ejemplo, **`MyTestSchedule`**.

   b. (Opcional) En Descripción, escriba una descripción para su programación. Por ejemplo, **`My first schedule`**.

   c. En Grupo de programaciones, elija un grupo de programaciones de la lista desplegable. Si no tiene un grupo, elija predeterminado. Para crear un grupo de programaciones, elija crear mi propia programación.

      Los grupos de programaciones se utilizan para agregar etiquetas a grupos de programaciones.

4. • Elija sus opciones de programación.

Ocurrencia	Haga lo siguiente...	
Programación única Una programación única invoca solo una vez un objetivo en la fecha y hora que especifique.	En Fecha y hora, realice lo siguiente: • Ingrese una fecha válida en el formato `YYYY/MM/DD` . • Ingrese una marca de tiempo en el formato `hh:mm` de 24 horas. • En Zona horaria, elija la zona horaria.	
Programación recurrente Una programación recurrente invoca un objetivo a una velocidad que especifique mediante una expresión cron o rate.	a. En Tipo de programación, realice una de las siguientes acciones: • Para utilizar una expresión Cron para definir la programac	

Ocurrencia	Haga lo siguiente...	
	ión, elija Programac ión basada en Cron e ingrese la expresión Cron. • Para utilizar una expresión de frecuenci a para definir la programación, elija Programación basada en la frecuencia e ingrese la expresión de frecuencia. Para obtener más información sobre las expresion es cron y rate, consulte Schedule types on EventBrid ge Scheduler en Amazon EventBridge Scheduler User Guide. b. En Intervalo de tiempo flexible, elija Apagado para desactivar la opción o elegir uno de los periodos de tiempo predefini dos. Por ejemplo, si elige 15 minutos y establece una programación recurrente para invocar su objetivo una vez cada hora,	

Ocurrencia	Haga lo siguiente...	
	el horario se ejecuta 15 minutos después del inicio de cada hora.	

5. (Opcional) Si elige Programación recurrente en el paso anterior, en la sección de Periodo de tiempo, realice lo siguiente:

   a. En Zona horaria, elija una zona horaria.

   b. En Fecha y hora de inicio, ingrese una fecha válida en el formato `YYYY/MM/DD` y, a continuación, especifique una marca de tiempo en el formato `hh:mm` de 24 horas.

   c. En Fecha y hora de finalización, ingrese una fecha válida en el formato `YYYY/MM/DD` y, a continuación, especifique una marca de tiempo en el formato `hh:mm` de 24 horas.

6. Elija Siguiente.

7. En la página Seleccionar destino, elija la operación de la API de AWS que invoca el Programador de EventBridge:

   a. Seleccione AWS Step FunctionsStartExecution.

   b. En la sección StartExecution, seleccione una máquina de estado o elija Crear nueva máquina de estado.

      En la actualidad, no se pueden ejecutar los flujos de trabajo rápidos sincrónicos según una programación.

   c. Introduzca una carga útil de JSON para la ejecución. Incluso si su máquina de estado no requiere ninguna carga útil JSON, debe incluir la entrada en formato JSON, como se muestra en el siguiente ejemplo.

```
{
    "Comment": "sampleJSONData"
}
```

8. Elija Siguiente.

9. En la página Configuración, haga lo siguiente:

   a. Para activar la programación, en Estado de la programación, cambie a Habilitar programación.

b. A fin de configurar una política de reintentos para su programación, en Política de reintento y cola de mensajes fallidos (DLQ), realice lo siguiente:

   - Cambie a Reintentar.
   - En Antigüedad máxima del evento, ingrese el máximo de horas y minutos que el Programador de EventBridge debe mantener un evento sin procesar.
   - El tiempo máximo es de 24 horas.
   - En Cantidad máxima de reintentos, ingrese el número máximo de veces que el Programador de EventBridge reintenta la programación si el objetivo devuelve un error.

     El valor máximo es 185 reintentos.

   Con las políticas de reintentos, si un programa no puede invocar su objetivo, el Programador de EventBridge vuelve a ejecutar el programa. Si se encuentra configurado, debe establecer el tiempo máximo de retención y los reintentos máximos para la programación.

c. Elija dónde almacena los eventos no entregados el Programador de EventBridge.

Opción de Cola de mensajes fallidos (DLQ)	Haga lo siguiente...
No almacenar	Elija None.
Guardar el evento en la misma Cuenta de AWS donde crea la programación	a. Elija Seleccionar una cola de Amazon SQS en mi Cuenta de AWS como DLQ. b. Elija el Nombre de recurso de Amazon (ARN) para la cola de Amazon SQS.

Opción de Cola de mensajes fallidos (DLQ)	Haga lo siguiente...	
Guardar el evento en una Cuenta de AWS diferente de donde crea la programación	a. Elija Especificar una cola de Amazon SQS en otras Cuentas de AWS como DLQ. b. Ingrese el Nombre de recurso de Amazon (ARN) para la cola de Amazon SQS.	

d. Para utilizar una clave administrada por el cliente a fin de cifrar la entrada de destino, en Cifrado, elija Personalizar la configuración de cifrado (avanzado).

   Si elige esta opción, ingrese un ARN de clave de KMS existente o elija Crear una AWS KMS key para navegar hasta la consola de AWS KMS. Para obtener más información sobre cómo el Programador de EventBridge cifra los datos en reposo, consulte Encryption at rest en Amazon EventBridge Scheduler User Guide.

e. Para que el Programador de EventBridge cree un rol de ejecución nuevo en su nombre, elija Crear un nuevo rol para esta programación. A continuación, ingrese un nombre para el Nombre de rol. Si elige esta opción, el Programador de EventBridge adjunta al rol los permisos necesarios para el objetivo creado con la plantilla.

10. Elija Siguiente.
11. En la página de Revisar y crear una programación, revise los detalles de su programación. En cada sección, elija Editar para volver a ese paso y editar sus detalles.
12. Elija Crear programación.

    Puede ver una lista de sus programaciones nuevas y existentes en la página de Programaciones. En la columna de Estado, verifique que su programación nueva se encuentre Habilitada.

Para confirmar que el Programador de EventBridge ha invocado la máquina de estado, compruebe los Registros de Amazon CloudWatch de la máquina de estado.

Recursos relacionados

Para obtener más información sobre el Programador de EventBridge, consulte lo siguiente:

- EventBridge Scheduler User Guide
- Referencia de la API del Programador de EventBridge
- Precios del Programador de EventBridge

Ejecuciones de flujos de trabajo estándar y rápidos en la consola

Al crear una nueva máquina de estado, debe seleccionar un Tipo que sea Estándar o bien Rápido. El Tipo predeterminado para las máquinas de estado es Estándar. Una máquina de estado cuyo Tipo es Estándar se denomina Flujo de trabajo estándar y una máquina de estado cuyo Tipo es Rápido se denomina Flujo de trabajo rápido.

Se define la máquina de estados mediante Lenguaje de estados de Amazon tanto para los flujos de trabajo estándar como rápidos. Las ejecuciones de la máquina de estado se comportarán de forma diferente en función del Type que seleccione.

> **Important**
>
> El Tipo que seleccione no se podrá cambiar después de que haya creado la máquina de estado.

Para obtener más información acerca de los flujos de trabajo estándar y rápidos, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

El historial de ejecuciones del flujo de trabajo estándar se registra en Step Functions, mientras que el historial de ejecuciones del flujo de trabajo rápido no se registra en Step Functions. Para registrar el historial de la ejecución de un flujo de trabajo de Express, debe configurarlo para enviar registros a Amazon CloudWatch. Para obtener más información, consulte Registro mediante CloudWatch Logs.

Una vez configurado el registro en un flujo de trabajo rápido, puede ver sus ejecuciones en la consola de Step Functions. La experiencia de la consola para ver las ejecuciones del flujo de trabajo rápido y las ejecuciones del flujo de trabajo estándar es similar, excepto por las siguientes diferencias y limitaciones.

> **Note**
>
> Como los datos de ejecución de los flujos de trabajo de Express se muestran mediante CloudWatch Logs Insights, el escaneo de los registros generará gastos. De forma predeterminada, el grupo de registros solo muestra las ejecuciones completadas en las últimas tres horas. Si se especifica un intervalo de tiempo mayor que incluya más eventos de ejecución, los costos aumentarán. Para obtener más información, consulte los registros vendidos en la pestaña Registros de la página de CloudWatch precios y. Registro mediante CloudWatch Logs

Contenido

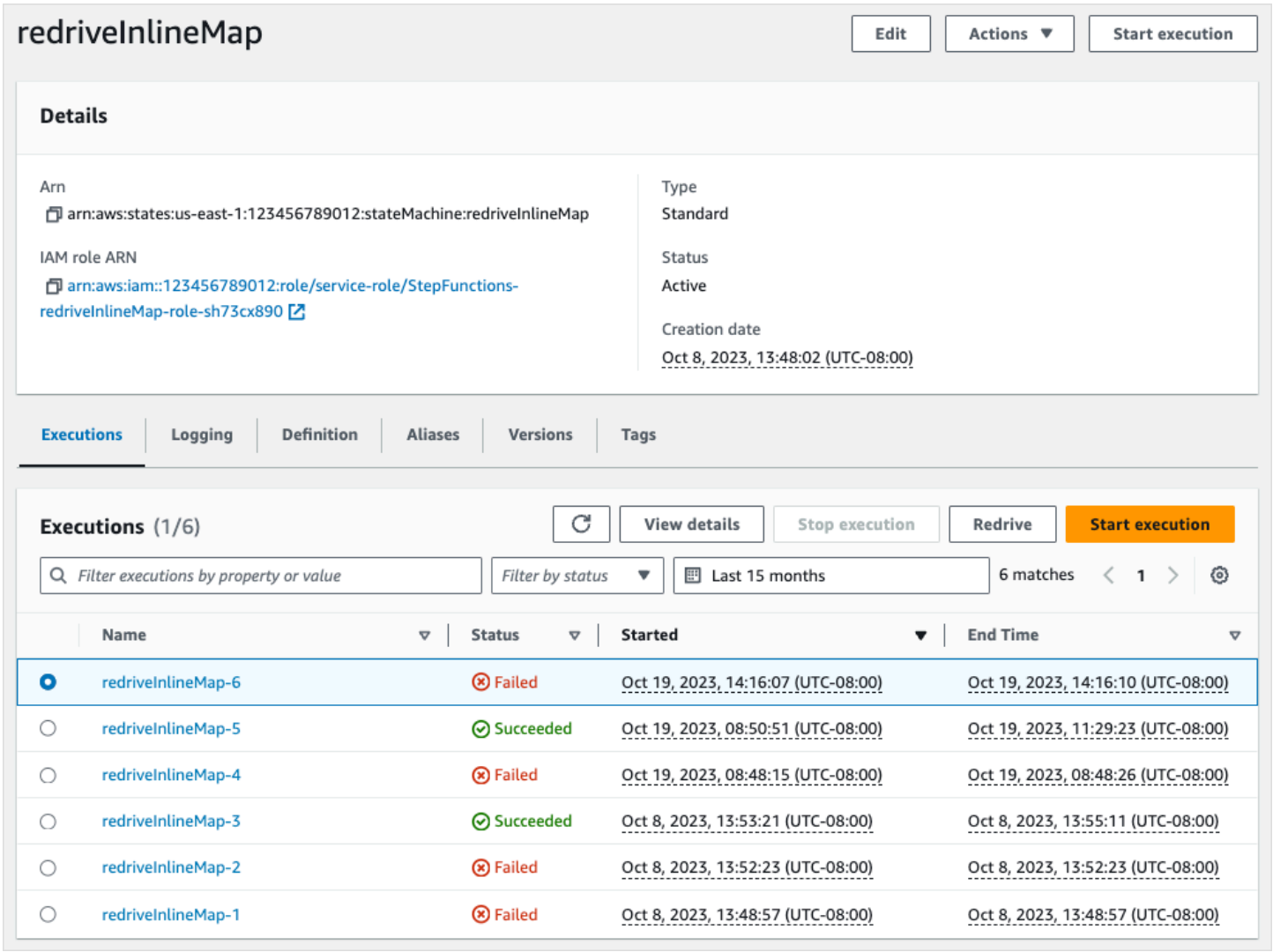
Diferencias de experiencia de consola

Para todos los flujos de trabajo estándar y rápidos, puede ver los detalles, como la máquina de estado y el ARN de su rol de IAM, en la página Detalles de la máquina de estado de la consola de Step Functions.

En la página Detalles de la máquina de estado, también puede ver una lista de los historiales de ejecución de la máquina de estado en la pestaña Ejecuciones. Utilice el cuadro Filtrar ejecuciones por propiedad o valor para buscar una ejecución específica, una versión o un alias de la máquina de estado elegida. Utilice el menú desplegable Todos para filtrar los historiales de ejecución por su estado. También puede elegir un historial de ejecución y pulsar el botón Ver detalles para abrir su página Detalles de la ejecución.

Flujos de trabajo estándar

Los historiales de ejecución de los flujos de trabajo estándar se encuentran siempre disponibles para las ejecuciones completadas en los últimos 90 días.



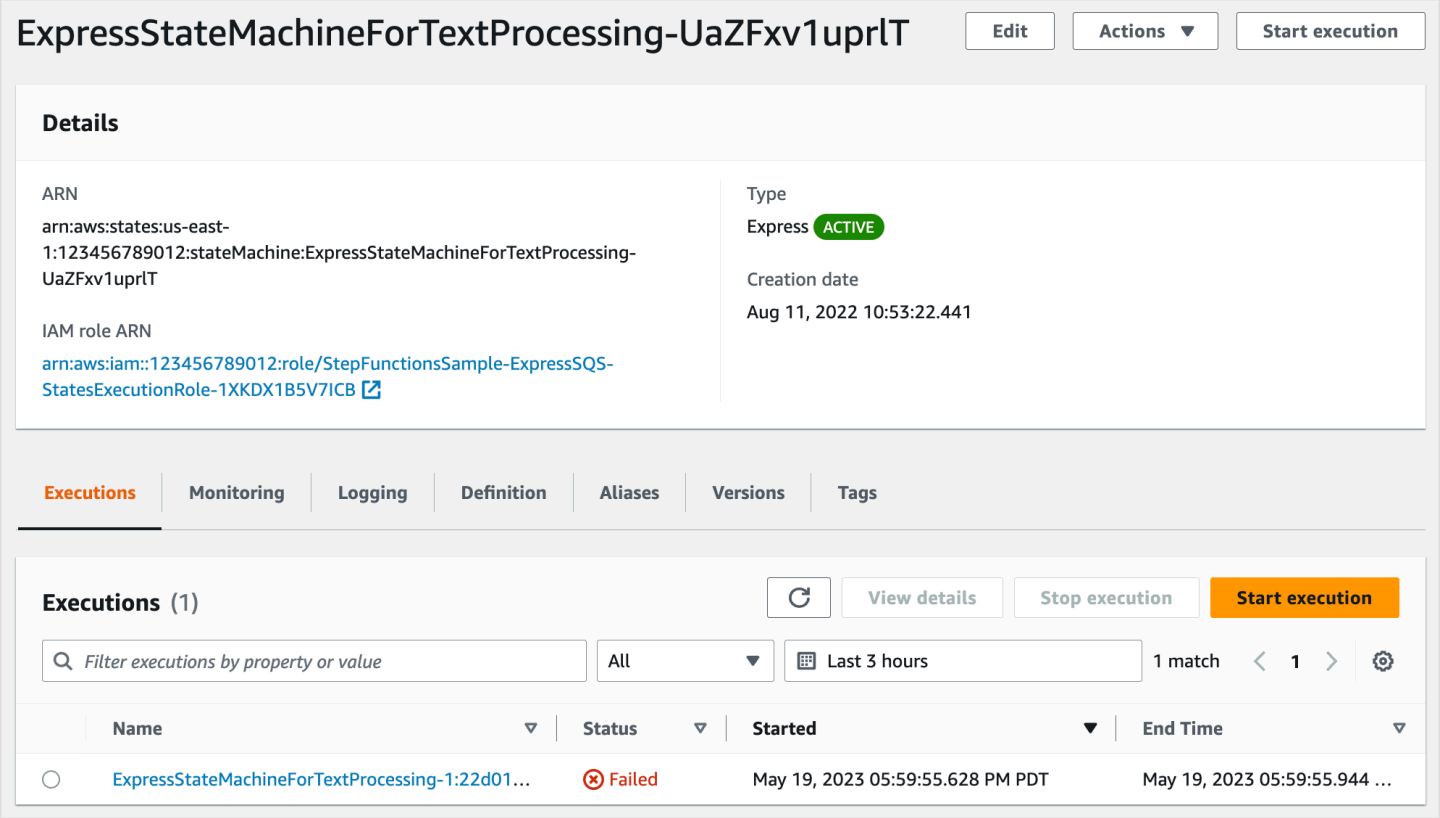
Flujos de trabajo rápidos

Para mostrar el historial de ejecución de los flujos de trabajo de Express, la consola Step Functions recupera los datos de registro recopilados a través de un grupo de CloudWatch registros.

También debe habilitar la nueva experiencia de consola para ver las ejecuciones de los flujos de trabajo rápidos. Para ello, pulse el botón Habilitar que aparece dentro del banner de la pestaña Ejecuciones. Una vez que seleccione este botón, no volverá a aparecer.

> **Tip**
>
> Para cambiar entre habilitar o deshabilitar la experiencia de consola, use el botón Habilitar historial de ejecución rápido.

Los historiales de las ejecuciones realizadas en las últimas tres horas están disponibles de forma predeterminada. Puede ajustar este intervalo de tiempo o especificar un intervalo personalizado. Si se especifica un intervalo de tiempo mayor que incluya más eventos de ejecución, el costo de analizar los registros aumentará. Para obtener más información, consulte los registros vendidos en la pestaña Registros de la página de CloudWatch precios y. Registro mediante CloudWatch Logs



Consideraciones y limitaciones para visualizar las ejecuciones de flujos de trabajo rápidos

Al visualizar las ejecuciones de flujos de trabajo rápidos en la consola de Step Functions, tenga en cuenta las siguientes consideraciones y limitaciones.

- La disponibilidad de los detalles de ejecución del flujo de trabajo de Express depende de Amazon CloudWatch Logs
- Los detalles parciales de la ejecución del flujo de trabajo rápido están disponibles si el nivel de registro es ERROR o FATAL
- La definición de la máquina de estado de una ejecución anterior no se puede ver una vez que se ha actualizado

La disponibilidad de los detalles de ejecución del flujo de trabajo de Express depende de Amazon CloudWatch Logs

> **Note**
>
> Si no habilita la nueva experiencia de consola para ver las ejecuciones del flujo de trabajo rápido, los historiales de ejecución y sus correspondientes detalles de ejecución no estarán disponibles en la consola de Step Functions. Para habilitar la nueva experiencia de consola, pulse el botón Habilitar que aparece dentro del banner de la pestaña Ejecuciones.

En el caso de los flujos de trabajo Express, su historial de ejecución y la información de ejecución detallada se recopilan a través de CloudWatch Logs Insights. Esta información se guarda en el grupo de CloudWatch registros que se especifica al crear la máquina de estados. El historial de ejecución de la máquina de estado se muestra en la pestaña Ejecuciones de la consola de Step Functions. La información detallada sobre cada ejecución de la máquina de estado se muestra en la página Detalles de la ejecución de la ejecución elegida.

> **Warning**
>
> Si elimina los CloudWatch registros de un flujo de trabajo rápido, no aparecerán en la pestaña Ejecuciones.

Recomendamos que se utilice el nivel de registro predeterminado de ALL para registrar todos los tipos de eventos de ejecución. Puede actualizar el nivel de registro según sea necesario para sus máquinas de estado existentes al editarlas. Para obtener más información, consulte Registro mediante CloudWatch Logs y Niveles de registro.

Los detalles parciales de la ejecución del flujo de trabajo rápido están disponibles si el nivel de registro es ERROR o FATAL

De forma predeterminada, el nivel de registro de las ejecuciones de flujos de trabajo rápidos se establece en ALL. Si se cambia el nivel de registro, los historiales de ejecución y los detalles de ejecución de las ejecuciones finalizadas no se verán afectados. No obstante, todas las ejecuciones nuevas emitirán registros según el nivel de registro actualizado. Para obtener más información, consulte Registro mediante CloudWatch Logs y Niveles de registro.

Por ejemplo, si cambia el nivel de registro de ALL a ERROR o FATAL, la pestaña Ejecuciones de la consola de Step Functions solo muestra las ejecuciones que producen error. En la pestaña Vista de eventos, la consola solo muestra los detalles del evento de la máquina de estado en los que se ha producido un error.

Recomendamos que se utilice el nivel de registro predeterminado de ALL para registrar todos los tipos de eventos de ejecución. Puede actualizar el nivel de registro según sea necesario para sus máquinas de estado existentes al editar la máquina de estado.

La definición de la máquina de estado de una ejecución anterior no se puede ver una vez que se ha actualizado

Las definiciones de máquinas de estado para ejecuciones anteriores no se almacenan en los flujos de trabajo rápidos. Si cambia la definición de la máquina de estado, solo podrá ver la definición de la máquina de estado para las ejecuciones que utilicen la definición más reciente.

Por ejemplo, si elimina uno o varios pasos de la definición de la máquina de estado, Step Functions detecta una discrepancia entre la definición y los eventos de ejecución anteriores. Puesto que las definiciones anteriores no se almacenan para los flujos de trabajo rápidos, Step Functions no puede mostrar la definición de la máquina de estado para las ejecuciones que se realicen en una versión anterior de la definición de la máquina de estado. Como resultado, las pestañas de Entrada y salida de ejecución, Definición, Vista de gráfico y Vista de tabla no están disponibles para las ejecuciones realizadas en versiones anteriores de una definición de máquina de estado.

Visualización y depuración de ejecuciones en la consola de Step Functions

La página Detalles de ejecución de la consola de Step Functions presenta información sobre las ejecuciones de máquinas de estado pasadas y en curso para los flujos de trabajo estándar y rápido. Esta información se muestra en formato de panel. Por ejemplo, puede encontrar la definición de Amazon States Language de la máquina de estado, su estado de ejecución, su ARN y el número total de transiciones de estado. También puede ver los detalles de ejecución de cualquier estado individual en la máquina de estado.

Contenido

- Detalles del paso
- Eventos
- Tutorial: Examinar las ejecuciones de máquinas de estados mediante la consola de Step Functions
  - Paso 1: Crear y probar las funciones de Lambda requerida
  - Paso 2: Crear y ejecutar la máquina de estado
  - Paso 3: Visualizar los detalles de la ejecución de la máquina de estado
  - Paso 4: Explorar los diferentes modos de visualización

Página de detalles de ejecución: información general de la interfaz

Puede encontrar los detalles de todas las ejecuciones de máquinas de estado en curso y pasadas, tanto para los flujos de trabajo estándar como para los rápidos, en la página Detalles de ejecución. Si especificó un identificador de ejecución al iniciar la ejecución, esta página se titulará con ese identificador de ejecución. De lo contrario, se titulará con el identificador de ejecución único que Step Functions genera automáticamente.

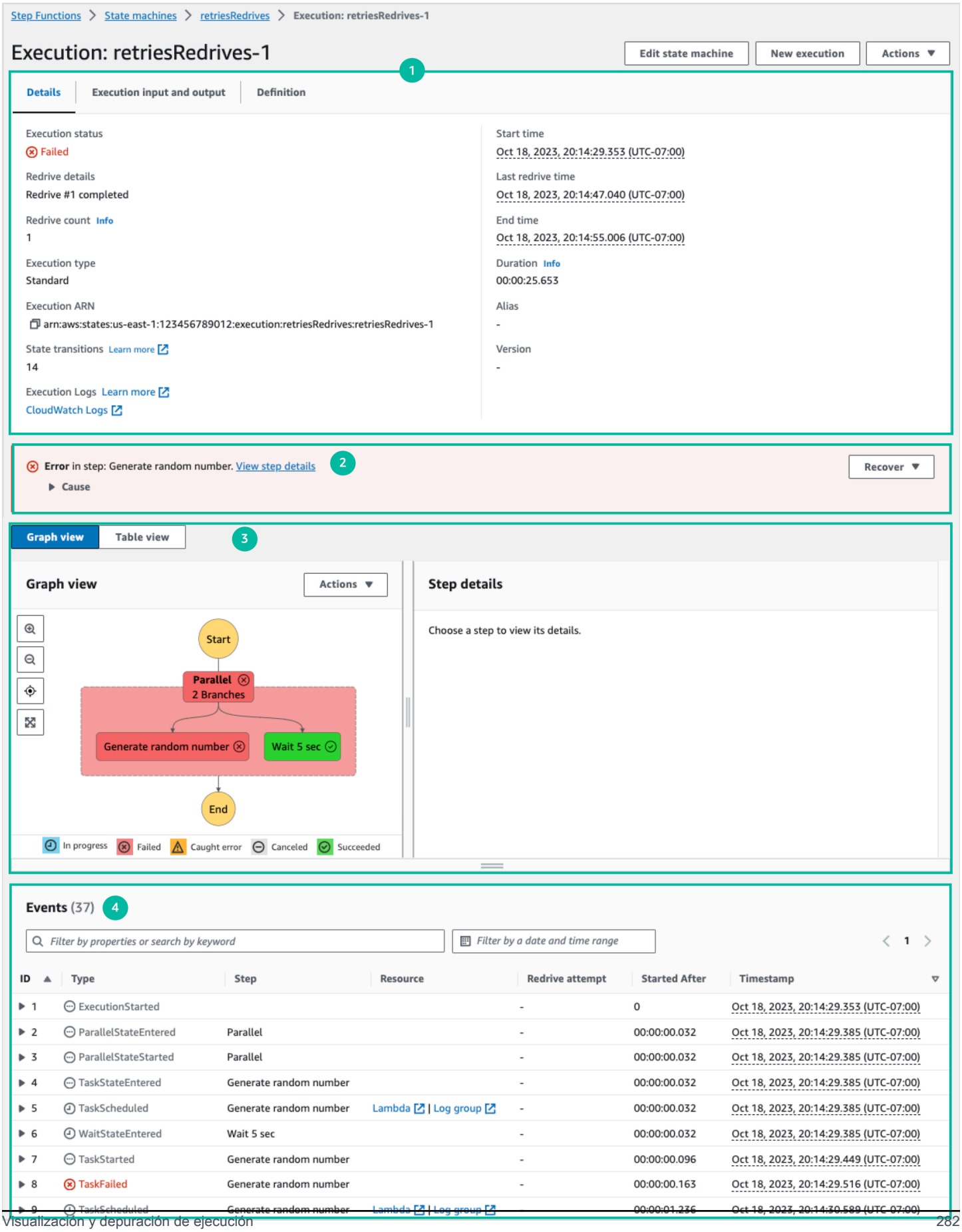
Además de las métricas de ejecución, la página Detalles de ejecución proporciona las siguientes opciones para gestionar la máquina de estado y su ejecución:

Button	Elija este botón para:
Editar máquina de estado	Editar la definición de Amazon States Language de la máquina de estado.
Nueva ejecución	Comenzar una nueva ejecución de la máquina de estado.
Acciones	Ofrece las siguientes opciones entre las que elegir: • Detener la ejecución: detiene una ejecución en curso. Esta opción no está disponible para las ejecuciones finalizadas. • Redrive: utilizar Redrive con ejecuciones de flujos de trabajo estándar que no se hayan completado correctamente en los últimos

Button	Elija este botón para:
	14 días. Estas incluyen las ejecuciones con error, anuladas o que hayan agotado su tiempo de espera. Para obtener más información, consulte Redriving de ejecuciones. • Exportar: exportar los detalles de la ejecución en formato JSON para compartirlos con otras personas o realizar análisis sin conexión a Internet. • Enviar comentarios: compartir comentarios sobre la interfaz.

> **Visualización de las ejecuciones iniciadas con una versión o alias**
>
> También puede ver las ejecuciones iniciadas con una versión o un alias en la consola de Step Functions. Para obtener más información, consulte Listado de ejecuciones por versiones y alias.

La página de la consola Detalles de ejecución contiene las siguientes secciones:


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. Resumen ejecutivo
2. Mensaje de error
3. Modo de visualización
4. Detalles del paso
5. Eventos

Resumen ejecutivo

La sección Resumen de ejecución aparece en la parte superior de la página Detalles de ejecución. Esta sección proporciona información general de los detalles de ejecución del flujo de trabajo. Esta información se divide en las tres pestañas siguientes:

Detalles

Muestra información, como el estado de la ejecución, el ARN y las marcas de tiempo de la hora de inicio y finalización de la ejecución. También puede ver el recuento total de las Transiciones de estado que se produjeron durante la ejecución de la máquina de estado. También puede ver los enlaces al mapa de rastreo de X-Ray y a los registros de Amazon CloudWatch Execution si ha activado el rastreo o los registros para su máquina de estados.

Si la ejecución de su máquina de estado la inició otra máquina de estado, puede ver el enlace de la máquina de estado principal en esta pestaña.

Si con la ejecución de la máquina de estado se utilizó redriven, en esta pestaña se muestra información relacionada con redrive, por ejemplo, el recuento de Redrive.

Entrada y salida de ejecución

Muestra la entrada y salida side-by-side de la ejecución de la máquina de estados.

Definición

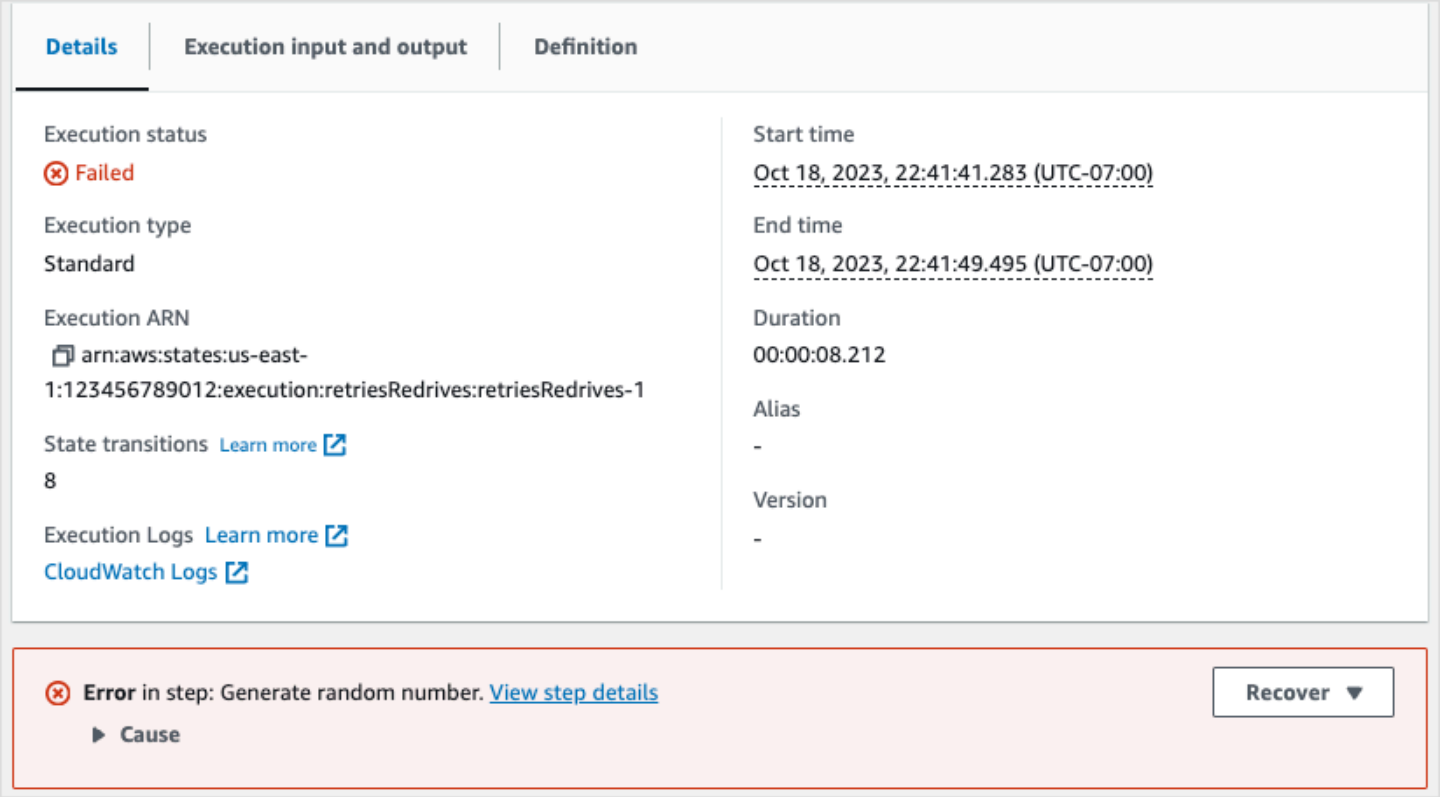
La definición de Amazon States Language de la máquina de estado.

Mensaje de error

Si la ejecución de la máquina de estado ha producido un error, la página Detalles de ejecución muestra un mensaje de error. Elija Causa o Ver detalles del paso en el mensaje de error para ver el motivo del error de ejecución o el paso que lo provocó.

Si se selecciona Ver detalles del paso, Step Functions resalta el paso que provocó el error en las pestañas Detalles del paso, Vista de gráfico y Vista de tabla. Si el paso es un estado Tarea, Map o Parallel para el que se hayan definido reintentos, el panel Detalles del paso muestra la pestaña Reintentar del paso. Además, si ha utilizado redriven con la ejecución, puede ver los reintentos y los detalles de ejecución de redrive en la pestaña Reintentos y redrives del Panel de detalles del paso.

Desde el botón desplegable Recuperar de este mensaje de error, puede utilizar redrive con las ejecuciones fallidas o iniciar una nueva ejecución. Para obtener más información, consulte Redriving de ejecuciones.



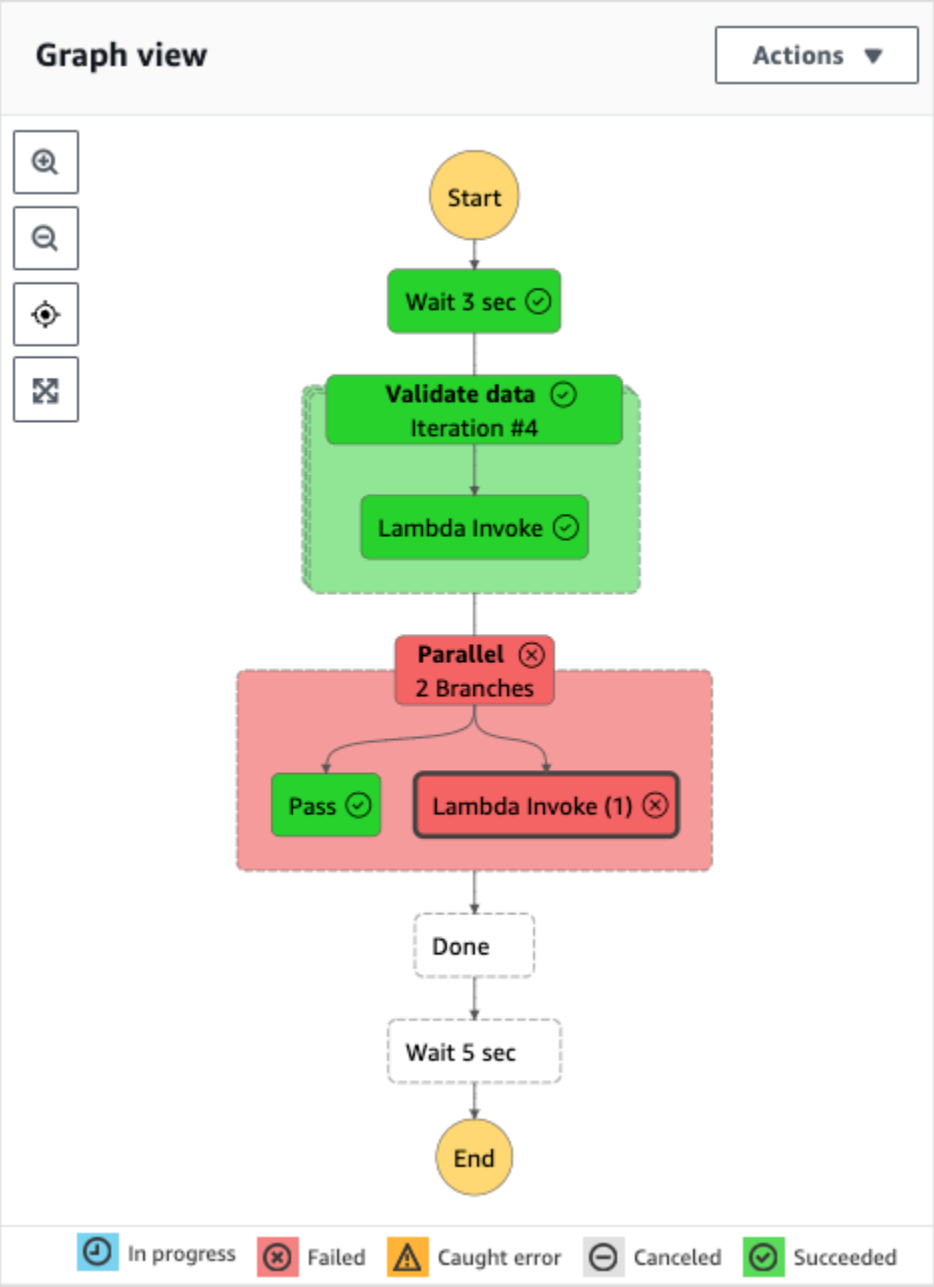
Modo de visualización

La sección Modo de visualización contiene dos visualizaciones diferentes para la máquina de estado. Puede elegir ver una representación gráfica del flujo de trabajo, una tabla con los estados del flujo de trabajo o una lista de los eventos asociados a la ejecución de la máquina de estado:

> **Note**
>
> Elija una pestaña para ver su contenido.

Graph view

El modo de Vista de gráfico muestra una representación gráfica del flujo de trabajo. En la parte inferior se incluye una leyenda que indica el estado de ejecución de la máquina de estado. También contiene botones que permiten acercar, alejar, centrar y alinear todo el flujo de trabajo o ver el flujo de trabajo en modo de pantalla completa.



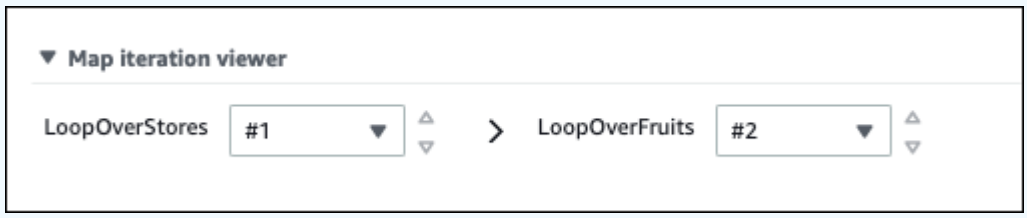
Desde esta vista, puede elegir cualquier paso del flujo de trabajo para ver los detalles relativos a su ejecución en el componente Detalles del paso. Al elegir un paso en la Vista de gráfico, la Vista

de tabla también muestra ese paso. Esto también es cierto a la inversa. Si elige un paso en la Vista de tabla, la Vista de gráfico también muestra el mismo paso.

Si la máquina de estado contiene un estado `Map`, un estado `Parallel` o ambos, puede ver sus nombres en el flujo de trabajo de la Vista de gráfico. Además, para el estado `Map`, la Vista de gráfico permite desplazarse por diferentes iteraciones de los datos de ejecución del estado Map. Por ejemplo, si el estado Map tiene cinco iteraciones y desea ver los datos de ejecución de la tercera y la cuarta iteraciones, haga lo siguiente:

1. Elija el estado Map cuyos datos de iteración desee ver.
2. En el visor de iteraciones de Map, elija #2 en la lista desplegable para la tercera iteración. Esto se debe a que las iteraciones se cuentan desde cero. Del mismo modo, elija #3 en la lista desplegable para la cuarta iteración del estado Map.

   También puede usar los controles
   ▵
   y
   ▿
   para moverse entre diferentes iteraciones del estado Map.

   > **Note**
   >
   > Si la máquina de estado contiene `Map` estados anidados, las listas desplegables de las iteraciones del estado `Map` principal y secundarias se mostrarán como se muestra en el siguiente ejemplo:
   >
   > 
   >

3. (Opcional) Si una o más de las iteraciones de estado de Map no se ejecutaron o la ejecución se detuvo, puede seleccionar esos números de iteración en Con error o Anulada en la lista desplegable.

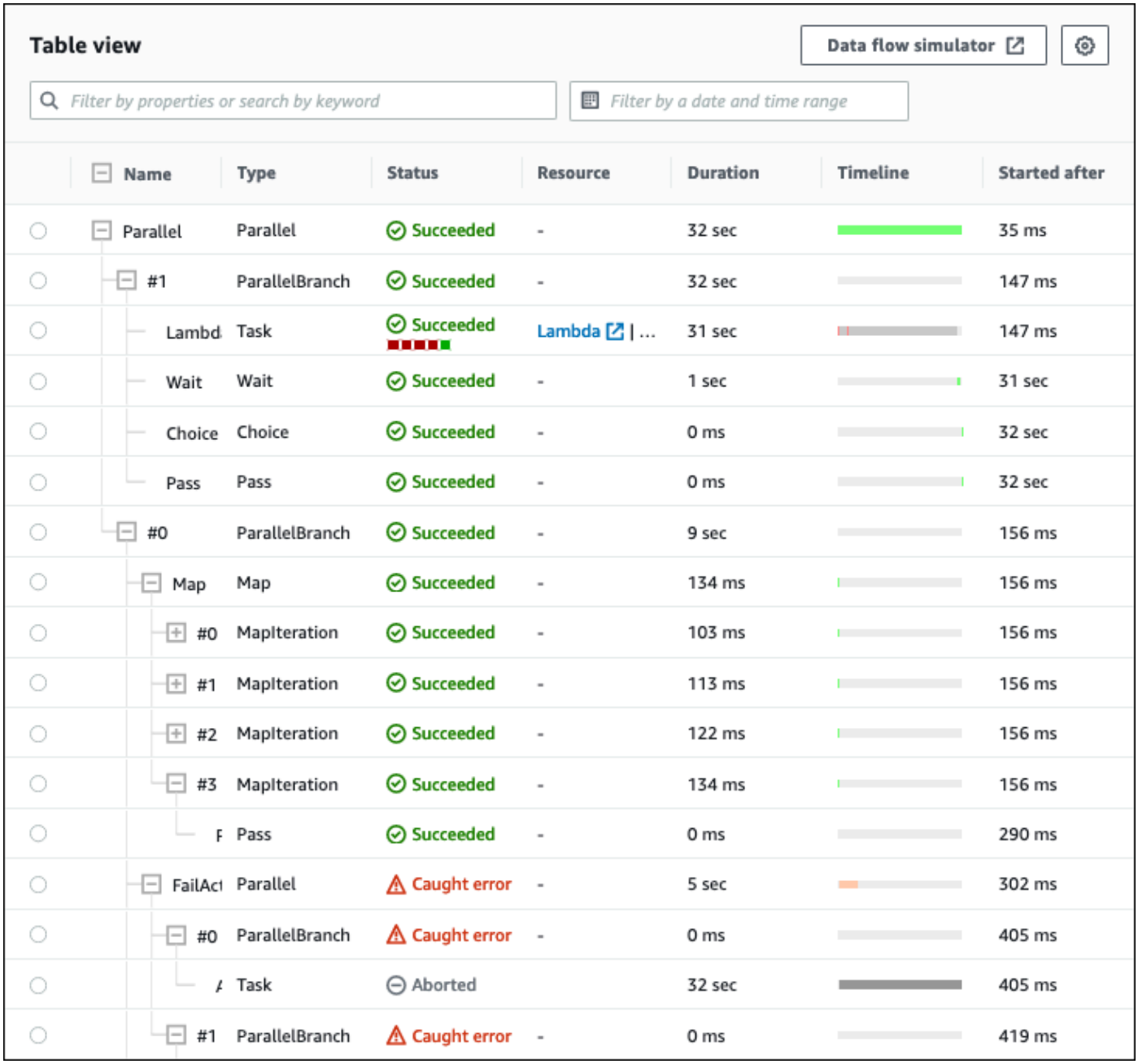
Por último, puede utilizar los botones Exportar y Diseño para exportar el gráfico del flujo de trabajo como una imagen SVG o PNG. También puede cambiar entre las vistas horizontal y vertical del flujo de trabajo.

Table view

El modo Vista de tabla muestra una representación tabular de los estados del flujo de trabajo. En este modo de visualización, puede ver los detalles de cada estado que se ejecutó en el flujo de trabajo, incluidos su nombre, el nombre de cualquier recurso que utilizó (como una AWS Lambda función) y si el estado se ejecutó correctamente.

Desde esta vista, puede elegir cualquier estado del flujo de trabajo para ver los detalles relativos a su ejecución en el componente Detalles del paso. Al elegir un paso en la Vista de tabla, la Vista de gráfico también muestra ese paso. Esto también es cierto a la inversa. Si elige un paso de en Vista de gráfico, la Vista de tabla muestra el mismo paso.

También puede limitar la cantidad de datos que se muestran en el modo de Vista de tabla mediante la aplicación de filtros a la vista. Puede crear un filtro para una propiedad específica, como estado o intento de Redrive. Para obtener más información, consulte Tutorial: Examinar las ejecuciones de máquinas de estados mediante la consola de Step Functions.

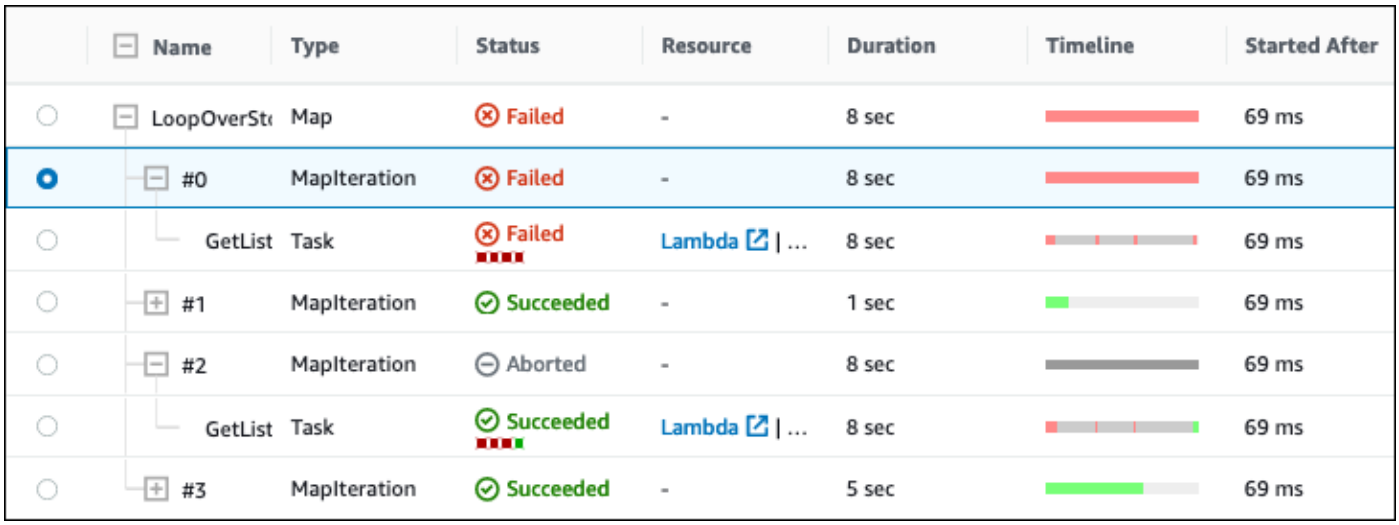


De forma predeterminada, este modo muestra las columnas Nombre, Tipo, Estado, Recurso e Iniciado después de. Puede configurar las columnas que desee ver mediante el cuadro de diálogo Preferencias. Las selecciones que realice en este cuadro de diálogo se conservarán para futuras ejecuciones de máquinas de estado hasta que se vuelvan a cambiar.

Si agrega la columna Plazo, la duración de la ejecución de cada estado se muestra con respecto al tiempo de ejecución de toda la ejecución. Se muestra como una línea de tiempo lineal codificada por colores. Esto puede ayudar a identificar cualquier problema de rendimiento relacionado con la ejecución de un estado específico. Los segmentos codificados por colores

para cada estado en el cronograma ayudan a identificar el estado de la ejecución, por ejemplo si está en curso, con error o anulada.

Por ejemplo, si ha definido reintentos de ejecución para un estado en su máquina de estado, estos reintentos se mostrarán en la línea temporal. Los segmentos rojos representan los intentos de `Retry` con error mientras que los segmentos gris claro representan el valor de `BackoffRate` entre intentos de `Retry`.



Si la máquina de estado contiene un estado **`Map`**, un estado **`Parallel`** o ambos, puede ver sus nombres en el flujo de trabajo en la Vista de tabla. Para estados `Map` y `Parallel`, el modo Vista de tabla muestra los datos de ejecución de sus iteraciones y ramas paralelas como nodos dentro de una vista de árbol. Puede elegir cada nodo de estos estados para ver sus detalles individuales en la sección Detalles del paso. Por ejemplo, puede revisar los datos de una iteración del estado Map específica que provocó el error del estado. Expanda el nodo del estado Map y, a continuación, consulte el estado de cada iteración en la columna Estado.

Detalles del paso

La sección Detalles del paso se abre a la derecha al elegir un estado en la Vista de gráfico o la Vista de tabla. Esta sección contiene las siguientes pestañas, que proporcionan información detallada sobre el estado seleccionado:

Entrada

Muestra los detalles de la entrada del estado seleccionado. Si hay un error en la entrada, se indica con

en el encabezado de la pestaña. Además, en esta pestaña se puede ver el motivo del error.

También puede elegir el botón de alternar Vista avanzada para ver la ruta de transferencia de los datos de entrada a medida que los datos pasan por el estado seleccionado. Esto permite identificar cómo se procesó la entrada a medida que se aplicaron a los datos uno o más campos, como `InputPath`, `Parameters` `ResultSelector`, `OutputPath` y `ResultPath`.

Salida

Muestra la salida del estado seleccionado. Si hay un error en la salida, se indica con

en el encabezado de la pestaña. Además, en esta pestaña puede ver el motivo del error.

También puede pulsar el botón de alternar Vista avanzada para ver la ruta de transferencia de datos de salida a medida que los datos pasan por el estado seleccionado. Esto permite identificar cómo se procesó la entrada a medida que se aplicaron a los datos uno o más campos, como `InputPath`, `Parameters` `ResultSelector`, `OutputPath` y `ResultPath`.

Detalles

Muestra información, como el tipo de estado, su estado de ejecución y la duración de la ejecución.

Para `Task` los estados que utilizan un recurso, por ejemplo AWS Lambda, esta pestaña proporciona enlaces a la página de definición del recurso y a la página de CloudWatch registros de Amazon para la invocación del recurso. También muestra los valores, si se especifican, para los campos `TimeoutSeconds` y `HeartbeatSeconds` del estado `Task`.

En el caso de los estados `Map`, esta pestaña muestra información sobre el recuento total de las iteraciones de un estado `Map`. Las iteraciones se clasifican como fallidas, abortadas, exitosas o. InProgress

Definición

Muestra la definición de Amazon States Language correspondiente al estado seleccionado.

Reintentar

> **Note**
>
> Esta pestaña aparece solo si ha definido un campo `Retry` en el estado `Task` o `Parallel` de la máquina de estado.

Muestra los reintentos iniciales y posteriores de un estado seleccionado en su intento de ejecución original. Para el intento fallido inicial y todos los intentos posteriores fallidos, elija ▶ junto a Tipo para ver el Motivo del error que aparece en un cuadro desplegable. Si el reintento se ha realizado correctamente, puede ver la Salida, que aparece en un cuadro desplegable.

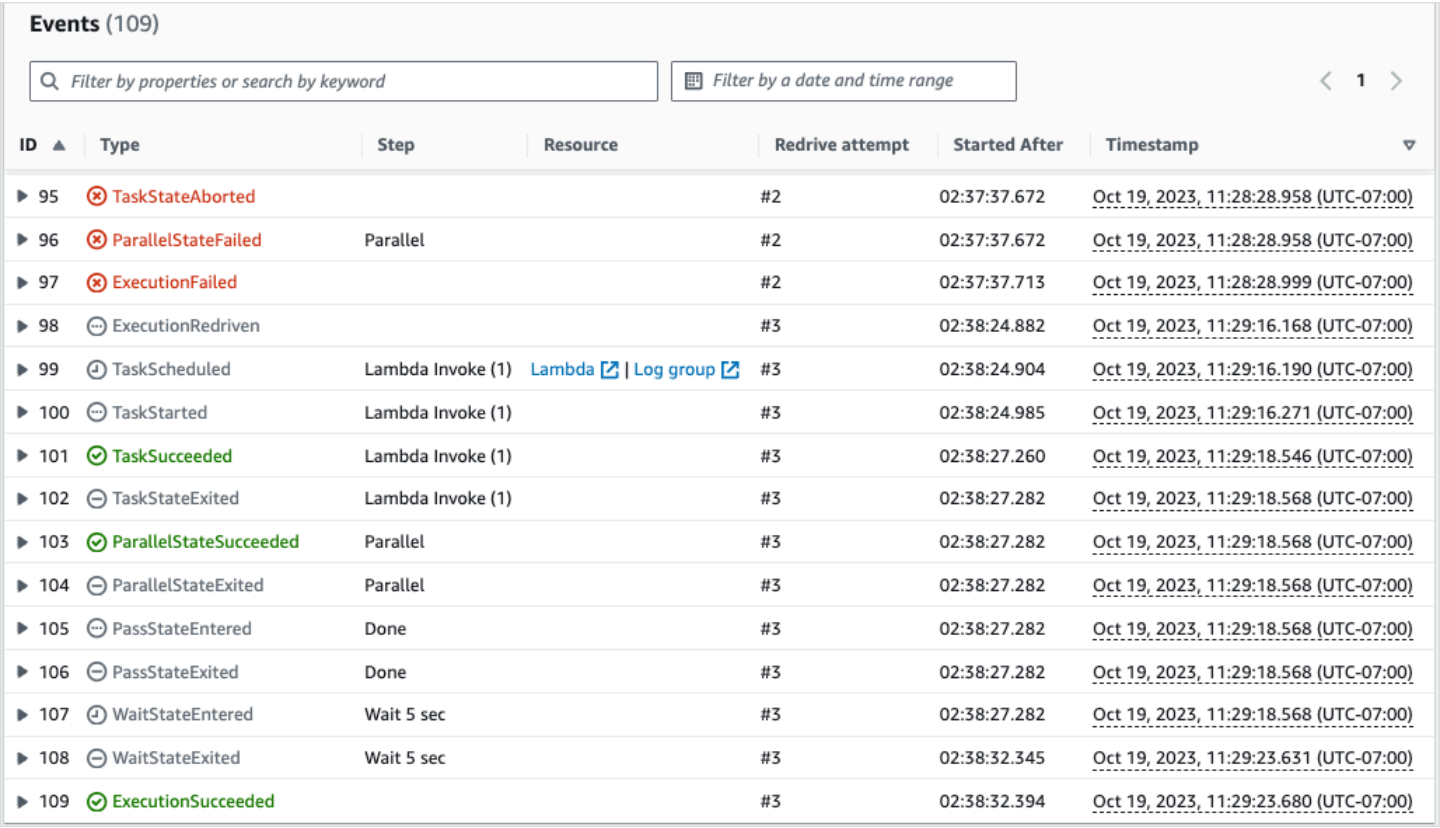
Si ha utilizado redriven con la ejecución, el encabezado de esta pestaña muestra el nombre Reintentos y redrives y muestra los detalles de reintento para cada redrive.

Eventos

Muestra una lista filtrada de los eventos asociados al estado seleccionado en una ejecución. La información que aparece en esta pestaña es un subconjunto del historial completo de eventos de ejecución que aparece en la tabla Eventos.

Eventos

La tabla Eventos muestra el historial completo de la ejecución seleccionada como una lista de eventos que abarca varias páginas. Cada página contiene hasta 25 eventos. En esta sección también se muestra el recuento total de eventos, que puede ayudar a determinar si se ha superado el recuento máximo del historial de eventos, que es de 25 000 eventos.



ID	Type	Step	Resource	Redrive attempt	Started After	Timestamp
95	TaskStateAborted			#2	02:37:37.672	Oct 19, 2023, 11:28:28.958 (UTC-07:00)
96	ParallelStateFailed	Parallel		#2	02:37:37.672	Oct 19, 2023, 11:28:28.958 (UTC-07:00)
97	ExecutionFailed			#2	02:37:37.713	Oct 19, 2023, 11:28:28.999 (UTC-07:00)
98	ExecutionRedriven			#3	02:38:24.882	Oct 19, 2023, 11:29:16.168 (UTC-07:00)
99	TaskScheduled	Lambda Invoke (1)	Lambda | Log group	#3	02:38:24.904	Oct 19, 2023, 11:29:16.190 (UTC-07:00)
100	TaskStarted	Lambda Invoke (1)		#3	02:38:24.985	Oct 19, 2023, 11:29:16.271 (UTC-07:00)
101	TaskSucceeded	Lambda Invoke (1)		#3	02:38:27.260	Oct 19, 2023, 11:29:18.546 (UTC-07:00)
102	TaskStateExited	Lambda Invoke (1)		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
103	ParallelStateSucceeded	Parallel		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
104	ParallelStateExited	Parallel		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
105	PassStateEntered	Done		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
106	PassStateExited	Done		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
107	WaitStateEntered	Wait 5 sec		#3	02:38:27.282	Oct 19, 2023, 11:29:18.568 (UTC-07:00)
108	WaitStateExited	Wait 5 sec		#3	02:38:32.345	Oct 19, 2023, 11:29:23.631 (UTC-07:00)
109	ExecutionSucceeded			#3	02:38:32.394	Oct 19, 2023, 11:29:23.680 (UTC-07:00)

De forma predeterminada, los resultados de la tabla Eventos se muestran en orden ascendente en función de la Marca temporal de los eventos. Puede cambiar la clasificación del historial de eventos de ejecución al orden descendente haciendo clic en el encabezado de la columna Marca temporal.

En la tabla Eventos, cada evento está codificado por colores para indicar su estado de ejecución. Por ejemplo, los eventos con error aparecen en rojo. Para ver detalles adicionales sobre un evento, seleccione

junto al ID del evento. Una vez abierto, los detalles del evento muestran la entrada, la salida y la invocación de recursos del evento.

Además, en la tabla Eventos, puede aplicar filtros para limitar los resultados del historial de eventos de ejecución que se muestran. Puede elegir propiedades como ID o Intento de Redrive. Para obtener más información, consulte Tutorial: Examinar las ejecuciones de máquinas de estados mediante la consola de Step Functions.

Tutorial: Examinar las ejecuciones de máquinas de estados mediante la consola de Step Functions

En este tutorial, aprenderá a inspeccionar la información de ejecución que se muestra en la página Detalles de la ejecución y a ver el motivo de una ejecución con error. A continuación, aprenderá a acceder a las distintas iteraciones de una ejecución de estado `Map`. Por último, aprenderá a configurar las columnas en la Vista de tabla y a aplicar los filtros adecuados para ver solo la información que le interese.

En este tutorial, creará una máquina de estado de tipo estándar que obtiene el precio de un conjunto de frutas. Para ello, la máquina de estados usa tres AWS Lambda funciones que devuelven una lista aleatoria de cuatro frutas, el precio de cada fruta y el costo promedio de las frutas. Las funciones de Lambda están diseñadas para arrojar un error si el precio de las frutas es inferior o igual a un valor de umbral.

> **Note**
>
> Aunque el siguiente procedimiento contiene instrucciones sobre cómo examinar los detalles de la ejecución de un flujo de trabajo estándar, también puede examinar los detalles de las ejecuciones de flujos de trabajo rápidos. Para obtener información acerca de las diferencias entre los detalles de la ejecución para tipos de flujos de trabajo estándar y rápidos, consulte Ejecuciones de flujos de trabajo estándar y rápidos en la consola.

Contenido

Paso 1: Crear y probar las funciones de Lambda requerida

1. Abra la consola de Lambda y, a continuación, realice los pasos 1 a 4 de la sección Paso 1: Crear una función de Lambda. Asegúrese de asignar un nombre a la función de Lambda **`GetListOfFruits`**.
2. Tras crear la función de Lambda, copie el Nombre de recurso de Amazon (ARN) de la función que aparece en la esquina superior derecha de la página. Para copiar el ARN, haga clic en  .

   El siguiente es un ejemplo de ARN, donde *`function-name`* es el nombre de la función de Lambda (en este caso, `GetListOfFruits`):

   ```
   arn:aws:lambda:us-east-1:123456789012:function:function-name
   ```
3. Copie el siguiente código de la función Lambda en el área Código fuente de la GetListOfFruitspágina.

   ```
   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. Seleccione Implementar y luego Probar para implementar los cambios y ver el resultado de la función de Lambda.
5. Cree dos funciones de Lambda adicionales, denominadas **`GetFruitPrice`** y **`CalculateAverage`** respectivamente, siguiendo estos pasos:

   a. Copie el siguiente código en el área Código fuente de la función GetFruitPriceLambda:

   ```
   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. Copie el siguiente código en el área Código fuente de la función CalculateAverageLambda:

   ```
   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. Asegúrese de copiar los ARN de estas dos funciones de Lambda y, a continuación, impleméntelas y pruébelas.

Paso 2: Crear y ejecutar la máquina de estado

Use la consola de Step Functions para crear una máquina de estado que invoque las funciones de Lambda que creó en el paso 1. En esta máquina de estado, se definen tres estados `Map`. Cada uno de estos estados `Map` contiene un estado `Task` que invoca una de las funciones de Lambda. Además, se define un campo `Retry` en cada estado `Task` con un número de reintentos definidos para cada estado. Si un estado `Task` detecta un error de tiempo de ejecución, se vuelve a ejecutar hasta el número de reintentos definido para ese estado `Task`.

1. Abra la consola de Step Functions y seleccione Escribir el flujo de trabajo en código.

> **Important**
>
> Asegúrese de que su máquina de estado esté en la misma AWS cuenta y región que la función Lambda que creó anteriormente.

2. En Tipo, mantenga la selección predeterminada de Estándar.
3. Copie la siguiente definición de Amazon States Language y péguela en Definición. Asegúrese de sustituir los ARN que se muestran por los de las funciones de Lambda que creó anteriormente.

```
{
    "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. Introduzca un nombre para la máquina de estado. Mantenga las selecciones predeterminadas para el resto de opciones de esta página y elija Crear máquina de estados.
5. Abra la página cuyo título es el nombre de su máquina de estado. Realice los pasos 1 a 4 de la sección Paso 4: Ejecutar la máquina de estado, pero use los siguientes datos como entrada de ejecución:

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
```

```
      "Store D"
    ]
}
```

Paso 3: Visualizar los detalles de la ejecución de la máquina de estado

En la página cuyo título es su ID de ejecución, puede revisar los resultados de la ejecución y depurar cualquier error.

1. (Opcional) Seleccione una de las pestañas que aparecen en la página Detalles de la ejecución para ver la información presente en cada una de ellas. Por ejemplo, para ver la entrada de la máquina de estado y su salida de ejecución, seleccione Entrada y salida de ejecución en la sección Resumen de ejecución.
2. Si la ejecución de la máquina de estado produce un error, seleccione Causa o Mostrar detalles del paso en el mensaje de error. Los detalles acerca del error aparecen en la sección Detalles del paso. Observe que el paso que provocó el error, que es el nombre de un `Task` estado GetListofFruits, aparece resaltado en la vista de gráfico y en la vista de tabla.

> **Note**
>
> Como el GetListofFruitspaso está definido dentro de un `Map` estado y no se ha podido ejecutar correctamente, el paso de **`Map`**estado se muestra como Fallido.

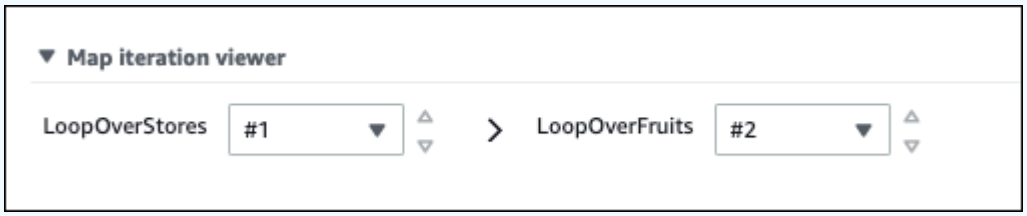
Paso 4: Explorar los diferentes modos de visualización

Puede elegir el modo que prefiera para ver el flujo de trabajo de la máquina de estado o el historial de eventos de ejecución. A continuación se indican algunas de las tareas que puede realizar en estos modos de visualización:

Vista de gráfico: cambiar entre diferentes iteraciones de estado **`Map`**

Si el estado Map tiene cinco iteraciones y desea ver los detalles de la ejecución de la tercera y la cuarta iteraciones, haga lo siguiente:

1. Seleccione el estado `Map` para el que desea ver los datos de iteración.
2. En el visor de iteraciones de Map, seleccione la iteración que desee ver. Las iteraciones se cuentan desde cero. Para elegir la tercera iteración de cinco, seleccione #2 en la lista desplegable que aparece junto al nombre del estado Map.

> **Note**
>
> Si la máquina de estado contiene estados `Map` anidados, Step Functions muestra las iteraciones de estado `Map` principal y secundaria en dos listas desplegables independientes:
>
> 
>

3. (Opcional) Si una o varias de las iteraciones de estado `Map` no se pudieron ejecutar o se detuvieron en un estado interrumpido, puede ver los detalles de la iteración con error. Para ver estos detalles, seleccione los números de iteración afectados en Con error o Interrumpido en la lista desplegable.

Vista de tabla: cambiar entre diferentes iteraciones de estado **`Map`**

Si el estado Map tiene cinco iteraciones y desea ver los detalles de la ejecución de las iteraciones número tres y cuatro, haga lo siguiente:

1. Seleccione el estado `Map` para el que desea ver los distintos datos de iteración.
2. En la vista en árbol de las iteraciones de estado `Map`, seleccione la fila de la iteración denominada #2 para la iteración número tres. Del mismo modo, elija la fila denominada #3 para la iteración número cuatro.

Vista de tabla: configurar las columnas que se van a mostrar

Elija

A continuación, en el cuadro de diálogo Preferencias, escoja las columnas que desee mostrar en Seleccionar las columnas visibles.

De forma predeterminada, este modo muestra las columnas Nombre, Tipo, Estado, Recurso e Iniciado después de.

Vista de tabla: filtrar los resultados

Limite la cantidad de información que se muestra aplicando uno o varios filtros basados en una propiedad, como Estado, o un intervalo de fechas y horas. Por ejemplo, para ver los pasos que no se ejecutaron correctamente, aplique el siguiente filtro:

1. Seleccione Filtrar por propiedades o buscar por palabra clave y, a continuación, Estado en Propiedades.
2. En Operadores, seleccione Estado =.
3. Seleccione Estado = Con error.
4. (Opcional) Seleccione Borrar filtros para eliminar los filtros aplicados.

Vista de eventos: filtrar los resultados

Limite la cantidad de información que se muestra aplicando uno o varios filtros basados en una propiedad, como Tipo, o un intervalo de fechas y horas. Por ejemplo, para ver los pasos del estado `Task` que no se ejecutaron correctamente, aplique el siguiente filtro:

1. Seleccione Filtrar por propiedades o buscar por palabra clave y, a continuación, Tipo en Propiedades.
2. En Operadores, seleccione Tipo =.
3. Elija Tipo = TaskFailed.
4. (Opcional) Seleccione Borrar filtros para eliminar los filtros aplicados.

Vista de eventos: inspecciona los detalles de un TaskFailedevento

Elija la

siguiente al ID de un TaskFailedevento para inspeccionar sus detalles, incluida la entrada, la salida y la invocación de recursos, que aparecen en un cuadro desplegable.

Redriving de ejecuciones

Puede utilizar redrive para reiniciar las ejecuciones de flujos de trabajo estándar que no se hayan completado correctamente en los últimos 14 días. Estas incluyen las ejecuciones con error, anuladas o que hayan agotado su tiempo de espera.

Cuando se utiliza redrive con una ejecución, se continúa con la ejecución fallida desde el paso con error y se utiliza la misma entrada. Step Functions conserva los resultados y el historial de ejecución de los pasos correctos y estos pasos no se vuelven a ejecutar cuando se realiza utiliza con redrive una ejecución. Por ejemplo, supongamos que el flujo de trabajo contiene dos estados: un estado Pass seguido de un estado Estado de la tarea. Si la ejecución del flujo de trabajo produce un error en el estado Task y se utiliza redrive con la ejecución, la ejecución se reprogramará y, a continuación, se volverá a ejecutar el estado Task.

Las ejecuciones de Redriven utilizan la misma definición de máquina de estado y el mismo ARN de ejecución que se usaron para el intento de ejecución original. Si el intento de ejecución original estaba asociado a una versión, un alias o ambos, la redriven ejecución se asociará a la misma versión, alias o a ambos. Incluso si se actualiza el alias para que apunte a una versión diferente, la ejecución de redriven seguirá utilizando la versión asociada al intento de ejecución original. Como las ejecuciones de redriven utilizan la misma definición de máquina de estados, deberá iniciar una nueva ejecución si actualiza la definición de máquina de estado.

Al utiliza redrive con una ejecución, el tiempo de espera a nivel de máquina de estado, si está definido, se restablece a 0. Para obtener más información acerca del tiempo de espera a nivel de máquina de estado, consulte `TimeoutSeconds`.

Las redrives de ejecución se consideran transiciones de estado. Para obtener información sobre cómo las transiciones de estado afectan a la facturación, consulte Precios de Step Functions.

Temas

Aptitud de Redrive para ejecuciones fallidas

Puede ejecutar redrive si el intento de ejecución original cumple las siguientes condiciones:

- Comenzó la ejecución el 15 de noviembre de 2023 o después de esa fecha. Las ejecuciones que haya iniciado antes de esta fecha no son aptas para redrive.
- El estado de la ejecución no es `SUCCEEDED`.
- La ejecución del flujo de trabajo no ha superado el período redrivable de 14 días. El período Redrivable se refiere al tiempo durante el cual se puede realizar redrive en una ejecución determinada. Este período comienza el día en que una máquina de estado completa su ejecución.
- La ejecución del flujo de trabajo no ha superado el tiempo máximo de apertura de un año. Para obtener información sobre las cuotas estatales de ejecución de máquinas, consulte Cuotas relacionadas con ejecuciones de máquinas de estado.
- El recuento del historial de eventos de ejecución es inferior a 24 999. Las ejecuciones Redriven añaden su historial de eventos al historial de eventos existente. Asegúrese de que la ejecución del flujo de trabajo contenga menos de 24 999 eventos para incluir el evento del historial de `ExecutionRedriven` y al menos otro evento del historial.

Comportamiento Redrive de los estados individuales

Según el estado en el que se produzca el error en el flujo de trabajo, el comportamiento redrive de todos los estados con error varía. La tabla siguiente describe el comportamiento de redrive para todos los estados.

Nombre de estado	Comportamiento de ejecución de Redrive
Pass	Si se produce un error en un paso anterior o se agota el tiempo de espera de la máquina de estado, el estado Pass se cierra y no se ejecuta en redrive.
Estado de la tarea	Programa y vuelve a iniciar el estado Task. Cuando se utiliza redrive con una ejecución que vuelve a ejecutar un estado Task, el valor de `TimeoutSeconds` del estado, si está definido, se restablece a 0. Para obtener más información acerca del tiempos de espera, consulte estado Tarea.

Nombre de estado	Comportamiento de ejecución de Redrive
Choice	Vuelve a evaluar las reglas del estado Choice.
Wait	Si el estado especifica `Timestamp` o `TimestampPath` que hace referencia a una marca de tiempo en el pasado, redrive provoca la salida del estado Wait y entra en el estado especificado en el campo `Next`.
Succeed	No utiliza redrive con las ejecuciones de máquina de estado que entran en el estado Succeed.
Fail	Vuelve a entrar en el estado Fail y a producir un error.
Parallel	Reprograma y utiliza redrives solo con las ramas que produjeron un error o se interrump ieron. Si el estado produjo un error debido a un error de `States.DataLimitExceeded`, se vuelve a ejecutar el estado Parallel, incluidas las ramas que se ejecutaron correctamente en el intento de ejecución original.
Estado Map en línea	Reprograma y utiliza redrives solo con las iteraciones que produjeron un error o se interrumpieron. Si el estado produjo un error debido a un error de `States.DataLimitExceeded`, se vuelve a ejecutar el estado Map en línea, incluidas las iteraciones que se ejecutaro n correctamente en el intento de ejecución original.

Nombre de estado	Comportamiento de ejecución de Redrive
Estado Map Distributed	Utiliza redrives con las ejecuciones con error del flujo de trabajo secundario en un Map Run. Para obtener más información, consulte Redriving de Map Runs. Si el estado produjo un error debido a un error de `States.DataLimitExceeded`, se vuelve a ejecutar el estado Map Distributed. Esto incluye los flujos de trabajo secundarios que se realizaron correctamente en el intento de ejecución original.

Permiso de IAM para utilizar redrive con una ejecución

Step Functions necesita el permiso adecuado para utilizar redrive con una ejecución. El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios para que la máquina de estado utilice redriving con una ejecución. Recuerde reemplazar el texto en *cursiva* por la información específica del recurso.

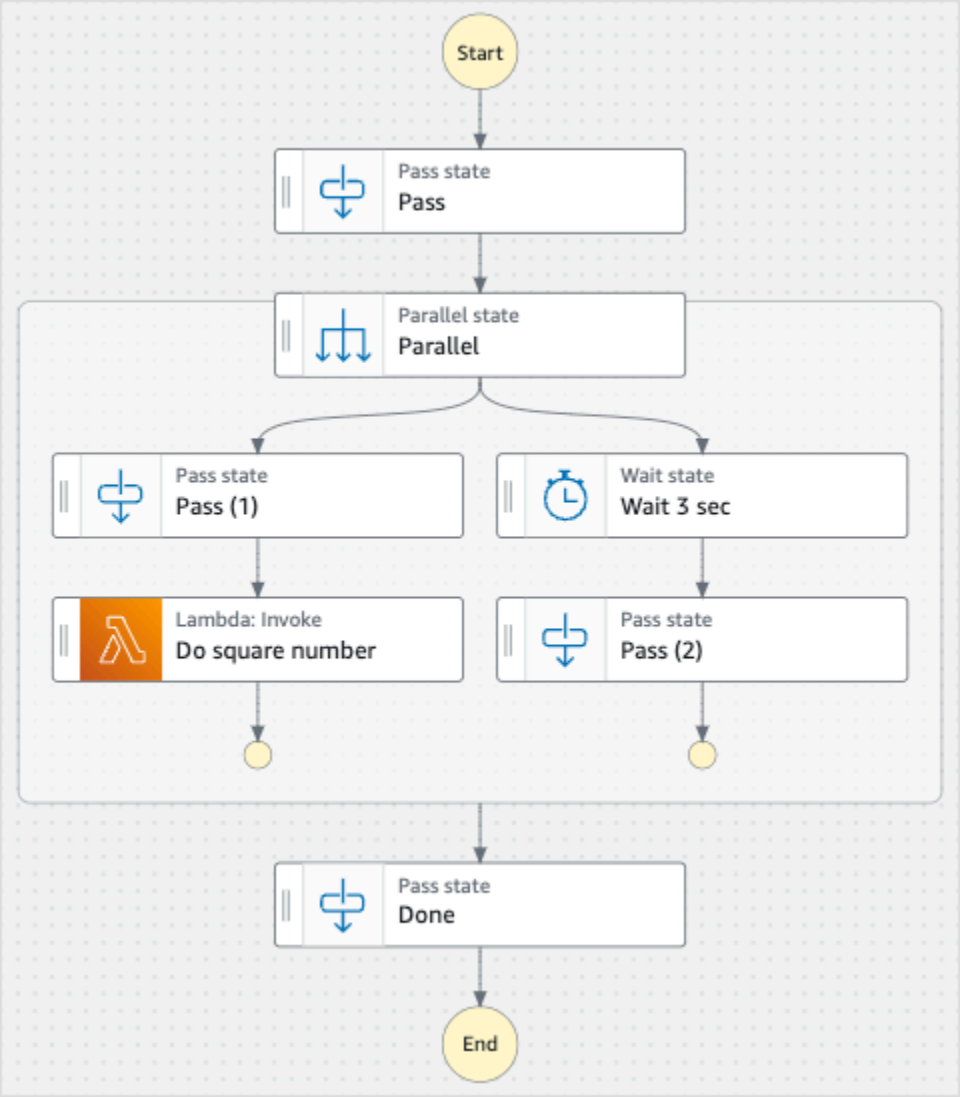
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

Para ver un ejemplo del permiso que necesita para utilizar redrive con un Map Run, consulte Ejemplo de política de IAM para redriving de estado Map Distributed.

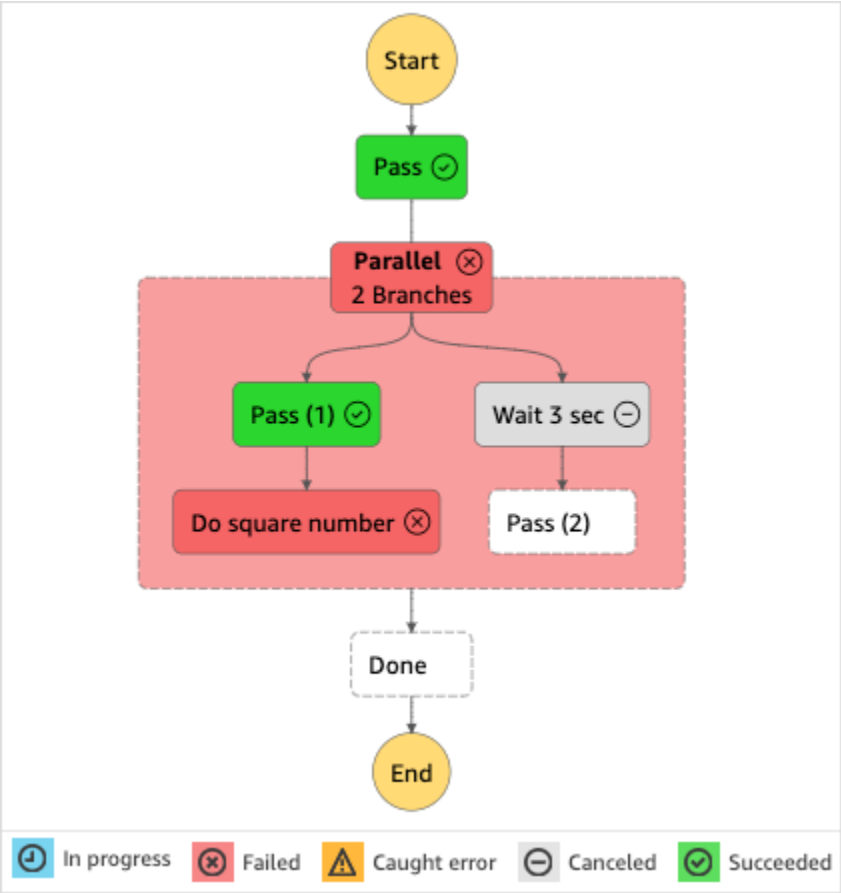
Redriving ejecuciones en consola

Puedes utilizar redrive con ejecuciones aptas desde la consola de Step Functions.

Por ejemplo, supongamos que la siguiente imagen representa el gráfico de flujo de trabajo de la máquina de estado.



Imagina que ejecuta esta máquina de estado. La siguiente imagen muestra el gráfico de la ejecución de la máquina de estado.



Como se muestra en esta imagen, el paso invoke Lambda denominado Do square number (elevar número al cuadrado) dentro del estado Parallel devolvió un error. Esto provocó un error en el estado Parallel. Las ramas cuya ejecución estaba en curso o no se había iniciado se detienen y se produce un error en la ejecución de la máquina de estado.

Para utilizar redrive con una ejecución desde la consola

1. Abra la Consola de Step Functions y, a continuación, elija una máquina de estado existente cuya ejecución no se haya completado correctamente.
2. En la página de detalles de la máquina de estado, en Ejecuciones, seleccione una instancia de ejecución fallida.
3. Elija Redrive.
4. En el cuadro de diálogo Redrive, elija Redrive ejecución.

> **Tip**
>
> Si se encuentra en la página de Detalles de la ejecución de una ejecución con error, lleve a cabo una de las siguientes acciones para utilizar redrive con la ejecución:
>
> - Elija Recuperar y, a continuación, seleccione una opción Redrive desde error.
> - Elija Acciones y, a continuación, seleccione Redrive.

Observe que redrive utiliza la misma definición de máquina de estado y el mismo ARN. Continúa la ejecución desde el paso que produjo un error en el intento de ejecución original. En este ejemplo, se trata de la rama Do square number (Elevar número al cuadrado) y Wait 3 seconds (Esperar 3 segundos) dentro del estado Parallel. Tras reiniciar la ejecución de estos pasos fallidos en el estado Parallel, redrive continuará con la ejecución del paso Done (Hecho).

5. Elija la ejecución, para abrir la página Detalles de la ejecución.

   En esta página, puede ver los resultados de la ejecución redriven. Por ejemplo, en la sección Resumen ejecutivo, puede ver el recuento de Redrive, que representa el número de veces que se ha utilizado redriven con una ejecución. En la sección Eventos, puede ver los eventos de ejecución relacionados con redrive adjuntos a los eventos del intento de ejecución original. Para ver un ejemplo, consulte `ExecutionRedriven`.

Redriving de ejecuciones mediante la API

Puede redrive optar a las ejecuciones mediante la RedriveExecutionAPI. Esta API reinicia las ejecuciones con error de los flujos de trabajo estándar a partir del paso con error, anulado o que agotó el tiempo de espera.

En el campo AWS Command Line Interface (AWS CLI), ejecuta el siguiente comando si la máquina redrive de estados no se ejecuta correctamente. Recuerde reemplazar el texto en *cursiva* por la información específica del recurso.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Examen de las ejecuciones redriven

Puede examinar una redriven ejecución en la consola o mediante las API: GetExecutionHistoryy DescribeExecution.

Examen de las ejecuciones redriven en la consola

1. Abra la consola de Step Functions y, a continuación, elija una máquina de estado existente en la que haya utilizado redriven con una ejecución.
2. Abra página Detalles de ejecución.

   En esta página, puede ver los resultados de la ejecución redriven. Por ejemplo, en la sección Resumen ejecutivo, puede ver el recuento de Redrive, que representa el número de veces que se ha utilizado redriven con una ejecución. En la sección Eventos, puede ver los eventos de ejecución relacionados con redrive adjuntos a los eventos del intento de ejecución original. Para ver un ejemplo, consulte `ExecutionRedriven`.

Examine las ejecuciones redriven mediante API

Si ha utilizado redriven con una ejecución de máquina de estado, puede usar una de las siguientes API para ver los detalles de la ejecución redriven. Recuerde reemplazar el texto en *`cursiva`* por la información específica del recurso.

- GetExecutionHistory — Devuelve el historial de la ejecución especificada como una lista de eventos. Esta API también devuelve los detalles sobre el intento de ejecución de redrive, si están disponibles.

  En el AWS CLI, ejecute el siguiente comando.

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution — Proporciona información sobre la ejecución de una máquina de estado. Puede ser la máquina de estado asociada a la ejecución, la entrada y salida de la ejecución, los detalles de la ejecución de redrive, si están disponibles, y los metadatos de ejecución relevantes.

  En AWS CLI, ejecute el siguiente comando.

```
aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Comportamiento de reintento de las ejecuciones redriven

Si la ejecución redriven vuelve a ejecutar un estado Estado de la tarea, Parallel o Map en línea para el que haya definido reintentos, el recuento de reintentos de estos estados se restablece a 0. Esto permite establecer el número máximo de intentos en redrive. Para una ejecución redriven, puede realizar un seguimiento de los reintentos individuales de estos estados mediante la consola.

Para examinar los reintentos individuales en la consola

1. En la página Detalles de ejecución de la consola de Step Functions, elija un estado que se haya reintentado en redrive.
2. Elija la pestaña Reintentos y redrives.
3. Elija
   ▶
   junto a cada reintento para ver sus detalles. Si el reintento se ha realizado correctamente, puede ver los resultados en la Salida, que aparece en un cuadro desplegable.

La siguiente imagen muestra un ejemplo de los reintentos realizados para un estado en el intento de ejecución original y en los redrives de esa ejecución. En esta imagen, se realizan tres reintentos en el intento original y en el de ejecución redrive. La ejecución se completa correctamente en el cuarto intento de redrive y devuelve un resultado de 16.

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

```
{
  "Squared": 16
}
```

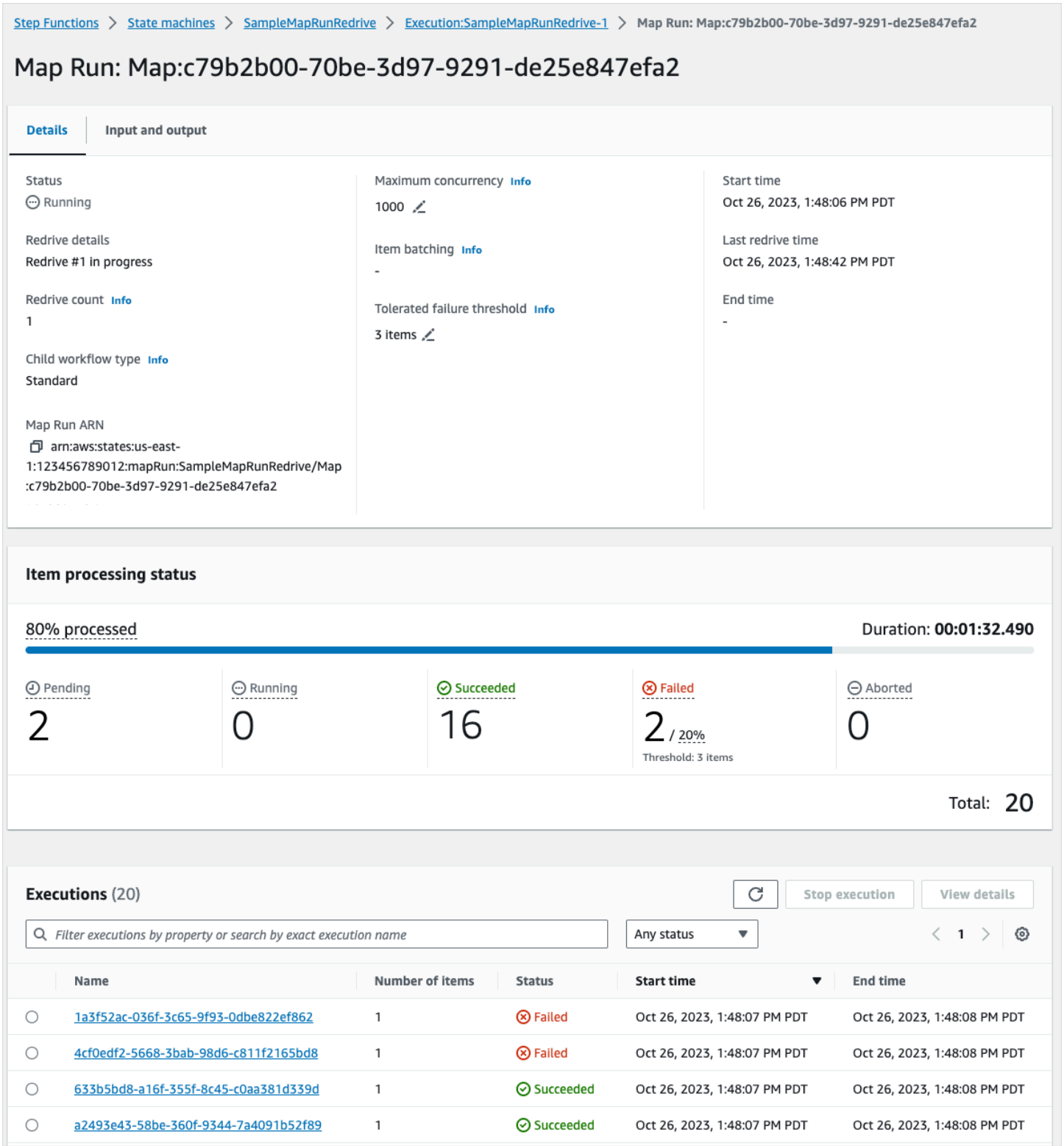
Formatted

Examen de Map Run de un estado Map Distributed

Cuando se ejecuta un estado `Map` en modo distribuido, Step Functions crea un recurso Map Run. Map Run hace referencia a un conjunto de ejecuciones de flujos de trabajo secundarios que inicia un estado Map Distributed y a la configuración de tiempo de ejecución que controla estas ejecuciones. Step Functions asigna un nombre de recurso de Amazon (ARN) al Map Run. Puede examinar un Map Run en la consola de Step Functions. También puede invocar la acción de la API `DescribeMapRun`. A Map Run también emite métricas a CloudWatch.

La consola de Step Functions proporciona una página Detalles de Map Run que muestra toda la información relacionada con la ejecución de un estado Map Distributed. Por ejemplo, puede ver el estado de la ejecución del estado Map Distributed, el ARN del Map Run y los estados de los elementos procesados en las ejecuciones del flujo de trabajo secundario iniciadas por el estado Map Distributed. También puede ver una lista de todas las ejecuciones de flujos de trabajo secundarios y acceder a sus detalles. Además, si el Map Run fue redriven, puede ver los detalles de redrive del Map Run en la sección Resumen de ejecución de Map Run. Por ejemplo, Última vez de redrive. La consola muestra esta información en formato de panel.

La página Detalles de Map Run contiene las siguientes secciones:



Contenido

- Estado del procesamiento de elementos
- Listado de ejecuciones
- Redriving de Map Runs
  - Aptitud de Redrive para los flujos de trabajo secundarios en un Map Run
  - Comportamiento de redrive en la ejecución de flujos de trabajo secundarios
  - Escenarios de entrada utilizados en Map Run redrive
  - Permiso de IAM para utilizar redrive con un Map Run
  - Redriving de Map Run en la consola
  - Redriving de Map Run mediante la API

Resumen de ejecución de Map Run

La sección Resumen de ejecución de Map Run aparece en la parte superior de la página Detalles de Map Run. Esta sección proporciona información general de los detalles de ejecución del Estado Map Distributed. Esta información se divide en las siguientes pestañas:

Detalles

Muestra información como el estado de ejecución del estado Map Distributed, el ARN del Map Run y el tipo de ejecuciones del flujo de trabajo secundario iniciadas por el estado Map Distributed. Puede ver configuraciones adicionales, como el umbral de error tolerado para el Map Run y la simultaneidad máxima especificada para las ejecuciones de flujos de trabajo secundarios. También puede editar estas configuraciones.

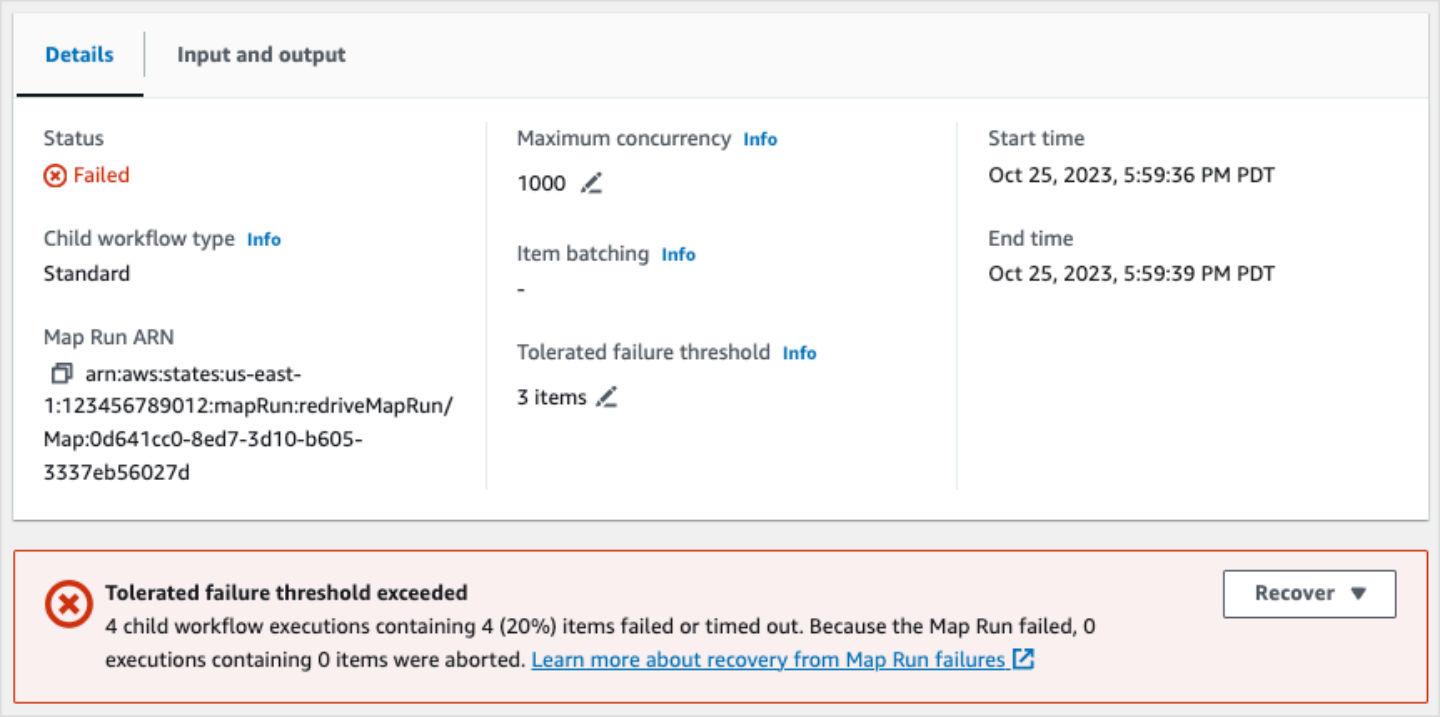
Entrada y salida

Muestra la entrada recibida por el estado Map Distributed y la salida correspondiente que genera. Por ejemplo, puede ver el conjunto de datos de entrada y su ubicación, así como los filtros de entrada aplicados a los elementos de datos individuales de ese conjunto de datos. Si exporta el resultado de la ejecución del estado Map Distributed, esta pestaña muestra la ruta al bucket de Amazon S3 que contiene los resultados de la ejecución. De lo contrario, lo dirigirá a la página Detalles de ejecución del flujo de trabajo principal para ver el resultado de la ejecución.

Mensaje de error

Si el Map Run produce un error, la página Detalles de Map Run muestra un mensaje de error con el motivo del error.

Desde el botón desplegable Recuperar de este mensaje de error, puede realizar redrive de las ejecuciones de flujo de trabajo secundario iniciadas por este Map Run o iniciar una nueva ejecución del flujo de trabajo principal. Para obtener más información, consulte Redriving de Map Runs.



Estado del procesamiento de elementos

La sección Estado del procesamiento de elementos muestra el estado de los elementos procesados en un Map Run. Por ejemplo, Pendiente indica que la ejecución de un flujo de trabajo secundario aún no ha empezado a procesar el elemento.

Los estados de los elementos dependen del estado de las ejecuciones de los flujos de trabajo secundarios que procesan los elementos. Si se produce un error en la ejecución de un flujo de trabajo secundario, se agota el tiempo de espera o un usuario cancela la ejecución, Step Functions no recibe ninguna información sobre el resultado del procesamiento de los elementos incluidos en la ejecución del flujo de trabajo secundario. Todos los elementos procesados por esa ejecución comparten el estado de ejecución del flujo de trabajo secundario.

Por ejemplo, supongamos que desea procesar 100 elementos en dos ejecuciones de flujos de trabajo secundarios, en las que cada ejecución procesa un lote de 50 elementos. Si una de las ejecuciones produce un error y la otra se realiza correctamente, tendrá 50 elementos correctos y 50 con error.

En la siguiente tabla se explican los tipos de estados de procesamiento disponibles para todos los elementos:

Estado	Descripción
Pendiente	Indica un elemento que la ejecución del flujo de trabajo secundario no ha empezado a procesar. Si un Map Run se detiene o produce un error, o un usuario cancela la ejecución antes de que comience el procesamiento de un elemento, el elemento permanece en estado Pendiente. Por ejemplo, si se produce un error en un Map Run con 10 elementos pendientes de procesar, estos 10 elementos permanecen en el estado Pendiente.
En ejecución	Indica un elemento que se está procesando actualmente en la ejecución del flujo de trabajo secundario.
Correcto	Indica que la ejecución del flujo de trabajo secundario procesó correctamente el elemento. Una ejecución correcta de un flujo de trabajo secundario no puede tener ningún elemento con error. Si un elemento del conjunto de datos falla durante la ejecución, se produce un error en la ejecución de todo el flujo de trabajo secundario.
Con error	Indica que la ejecución del flujo de trabajo secundario no pudo procesar el elemento o se agotó el tiempo de espera de la ejecución . Si se produce un error en alguno de los elementos procesados por la ejecución de un flujo de trabajo secundario, se produce un

Estado	Descripción
	error en toda la ejecución del flujo de trabajo secundario. Por ejemplo, consideremos la ejecución de un flujo de trabajo secundario que procese 1000 elementos. Si algún elemento de ese conjunto de datos produce un error durante la ejecución , Step Functions considera que se ha producido un error en toda la ejecución del flujo de trabajo secundario. Al utilizar redrive con un Map Run, el recuento de elementos con este estado se restablece a 0.

Estado	Descripción
Anulado	Indica que la ejecución del flujo de trabajo secundario comenzó a procesar el elemento, pero el usuario canceló la ejecución o Step Functions detuvo la ejecución porque se produjo un error al ejecutar el mapa. Por ejemplo, pensemos en la ejecución de un flujo de trabajo secundario En ejecución que procese 50 elementos. Si el Map Run se detiene debido a un error o a que un usuario canceló la ejecución, la ejecución del flujo de trabajo secundario y el estado de los 50 elementos cambiarán a Anulado. Si utiliza una ejecución de flujo de trabajo secundario de tipo Rápido, no podrá detener la ejecución. Cuando se utiliza redrive con una Map Run que inicia ejecuciones de flujos de trabajo secundarios de tipo rápido, el recuento de elementos con este estado se restablece a 0. Esto se debe a que los flujos de trabajo secundarios de Express se reinician mediante la acción de la StartExecutionAPI en lugar de hacerlo. redriven

Listado de ejecuciones

La sección Ejecuciones muestra todas las ejecuciones del flujo de trabajo secundario de un Map Run específico. Utilice el campo Buscar por nombre exacto de ejecución para buscar una ejecución de flujo de trabajo secundario específico. También puede usar el menú desplegable Cualquier estado para filtrar los historiales de ejecución de flujo de trabajo secundario por su estado. Para ver los detalles de una ejecución específica, seleccione una ejecución de flujo de trabajo secundario de la lista y pulse el botón Ver detalles para abrir su página Detalles de ejecución.

Important

La política de retención para las ejecuciones de flujos de trabajo secundarios es de 90 días. Las ejecuciones de flujos de trabajo secundarios finalizadas que tengan una antigüedad superior a este período de retención no se mostrarán en la tabla Ejecuciones. Esto es cierto incluso si el estado Map Distributed o el flujo de trabajo principal siguen ejecutándose durante más tiempo que el período de retención. Puede ver los detalles de ejecución, incluidos los resultados, de estas ejecuciones de flujos de trabajo secundarios si exporta la salida del estado Map Distributed a un bucket de Amazon S3 mediante`ResultWriter`.

Tip

Pulse el botón de actualización

para ver la lista más actualizada de todas las ejecuciones de flujos de trabajo secundarios.

Redriving de Map Runs

Puede reiniciar las ejecuciones fallidas de flujos de trabajo secundarios en un flujo de trabajo basado en Map Run mediante redriving del flujo de trabajo principal. Un flujo de trabajo principal redriven redrives todos los estados fallidos, incluido el estado Map Distributed. Un flujo de trabajo principal reconduce los estados de error si no hay ningún evento `<stateType>Exited` correspondiente al evento `<stateType>Entered` para un estado cuando el flujo de trabajo principal completó su ejecución. Por ejemplo, si el historial de eventos no contiene el evento `MapStateExited` de un evento `MapStateEntered`, puede utilizar redrive con el flujo de trabajo principal para utilizar redrive con todas las ejecuciones fallidas de flujos de trabajo secundarios en el Map Run.

Un Map Run no se inicia o produce un error en el intento de ejecución original cuando la máquina de estado no tiene el permiso necesario para acceder a ItemReader, a ResultWriter o a ambas. Si el Run Map no se inició en el intento de ejecución original del flujo de trabajo principal, el uso de redriving con el flujo de trabajo principal inicia la ejecución del Map Run por primera vez. Para solucionar este problema, añada los permisos necesarios al rol de la máquina de estado y, a continuación, utilice redrive con el flujo de trabajo principal. Si utiliza redrive con el flujo de trabajo principal sin añadir los permisos necesarios, intentará iniciar una nueva ejecución de Map Run que

volverá a producir un error. Para obtener información acerca de los permisos que podría necesitar, consulte Políticas de IAM para usar el estado Map Distributed.

Temas

- Aptitud de Redrive para los flujos de trabajo secundarios en un Map Run
- Comportamiento de redrive en la ejecución de flujos de trabajo secundarios
- Escenarios de entrada utilizados en Map Run redrive
- Permiso de IAM para utilizar redrive con un Map Run
- Redriving de Map Run en la consola
- Redriving de Map Run mediante la API

Aptitud de Redrive para los flujos de trabajo secundarios en un Map Run

Puede utilizar sin éxito redrive en ejecuciones fallidas de flujos de trabajo secundario de Map Run si se cumplen las siguientes condiciones:

- Comenzó la ejecución del flujo de trabajo principal el 15 de noviembre de 2023 o después de esa fecha. Las ejecuciones que haya iniciado antes de esta fecha no son aptas para redrive.
- No ha superado el límite máximo de 1000 redrives de un Map Run dado. Si ha superado este límite, recibirá el error `States.Runtime`.
- El flujo de trabajo principal es redrivable. Si el flujo de trabajo principal no es redrivable, no podrá utilizar redrive con el flujo de trabajo secundario en un Map Run. Para obtener más información sobre la aptitud para redrive de un flujo de trabajo, consulte Aptitud de Redrive para ejecuciones fallidas.
- Las ejecuciones de flujos de trabajo secundarios del tipo estándar en el Map Run no han superado el límite del historial de eventos de ejecución de 25 000. Las ejecuciones de flujos de trabajo secundarios que superan el límite del historial de eventos se contabilizan para el umbral de error tolerado y se consideran fallidas. Para obtener más información acerca de la aptitud para redrive de una ejecución, consulte Aptitud de Redrive para ejecuciones fallidas.

En los siguientes casos, se inicia un nuevo Map Run y no se utiliza redriven con el Map Run existente aunque si la ejecución del Map Run falle en el intento de ejecución original:

- El Map Run falló debido al error de `States.DataLimitExceeded`.

- Error en Map Run debido a un error de interpolación de datos de JSON, `States.Runtime`. Por ejemplo, seleccionó un nodo JSON inexistente en OutputPath.

Un Map Run puede seguir ejecutándose incluso después de que el flujo de trabajo principal se detenga o agote el tiempo de espera. En estos escenarios, redrive no ocurre de forma inmediata:

- Es posible que Map Run aún esté cancelando ejecuciones de flujos de trabajo secundarios de tipo estándar en curso o esperando a que las ejecuciones de flujos de trabajo secundarios de tipo rápido completen sus ejecuciones.
- Es posible que Map Run siga escribiendo resultados en ResultWriter, si lo ha configurado para exportarlos.

En estos casos, el Map Run en ejecución completa sus operaciones antes de intentar utilizar redrive.

Comportamiento de redrive en la ejecución de flujos de trabajo secundarios

Las ejecuciones de flujos de trabajo secundarios de redriven en un Map Run muestran el comportamiento que se describe en la siguiente tabla.

Flujo de trabajo secundario rápido	Flujo de trabajo secundario estándar
Todas las ejecuciones del flujo de trabajo secundario que fallaron o agotaron el tiempo de espera en el intento de ejecución original se inician mediante la acción de la StartExecutionAPI. El primer estado en ItemProcessor se ejecuta primero.	Con todas las ejecuciones de flujos de trabajo secundarios que produzcan un error, agoten su tiempo de espera o se cancelen en el intento de ejecución original se utiliza redriven con la acción de la API RedriveExecution. Estos flujos redriven de trabajo secundarios pertenecen al último estado en el ItemProcessor que no se ejecutó correctamente.
Con las ejecuciones fallidas siempre se puede utilzar redriven. Esto se debe a que las ejecuciones de flujos de trabajo secundarios de Express siempre se inician como una nueva ejecución mediante la acción de la StartExecution API.	No siempre es posible utilizar redriven con los flujos de trabajo secundarios estándar. Si con una ejecución no se utiliza redrivable, no se vuelve a intentar. El último error o salida de la ejecución es permanente. Esto es posible cuando una ejecución supera los

Flujo de trabajo secundario rápido	Flujo de trabajo secundario estándar
	25 000 eventos históricos o cuando su período redrivable de 14 días ha caducado. Es posible que no se utilice redrivable con una ejecución de flujo de trabajo secundari o estándar si la ejecución del flujo de trabajo principal se ha cerrado en un plazo de 14 días, pero la ejecución del flujo de trabajo secundario se ha cerrado antes de 14 días.
Las ejecuciones de flujos de trabajo secundari os rápidos utilizan el mismo ARN de ejecución que el intento de ejecución original, pero no se puede identificar claramente cada uno de sus redrives.	Las ejecuciones de flujos de trabajo secundari os estándar utilizan el mismo ARN de ejecución que el intento de ejecución original. Puede identificar claramente a la persona redrives en la consola y mediante API, como GetExecut ionHistoryy DescribeExecution. Para obtener más información, consulte Examen de las ejecuciones redriven.

Si un Map Run ha sido redriven y ha alcanzado su límite de simultaneidad, las ejecuciones de flujos de trabajo secundarios de este Map Run pasarán al estado pendiente. El estado de ejecución de Map Run también pasa al estado Pendiente de redrive. Hasta que el límite de simultaneidad especificado permita ejecutar más flujos de trabajo secundarios, la ejecución permanecerá en el estado Pendiente de redrive.

Por ejemplo, supongamos que el límite de simultaneidad del estado Map Distributed en el flujo de trabajo es de 3000 y el número de flujos de trabajo secundarios que se van a volver a ejecutar es de 6000. Esto hace que 3000 flujos de trabajo secundarios se ejecuten en paralelo, mientras que los 3000 flujos de trabajo restantes permanezcan en el estado Pendientes de reconducción. Una vez que el primer lote de 3000 flujos de trabajo secundarios completa su ejecución, se ejecutan los 3000 flujos de trabajo secundarios restantes.

Cuando un Map Run finaliza su ejecución o se anula, el recuento de ejecuciones de flujos de trabajo secundarios en el estado Pendiente de redrive se restablece a 0.

Escenarios de entrada utilizados en Map Run redrive

En función de cómo se haya introducido la entrada en el estado Map Distributed en el intento de ejecución original, una ejecución de Map Run redriven utilizará la entrada tal y como se describe en la siguiente tabla.

Entrada en el intento de ejecución original	Entrada utilizada en Map Run redrive
Entrada transferida desde un estado anterior o desde la entrada de ejecución.	El Map Run redriven utiliza la misma entrada.
La entrada transferida mediante ItemReader y Map Run no inició las ejecuciones del flujo de trabajo secundario porque se cumple una de las siguientes condiciones: • Map Run falló y se produjo el error `States.ItemReaderFailed`. • Map Run falló y se produjo el error `States.ResultWriterFailed`. • Se agotó el tiempo de espera de la ejecución del flujo de trabajo principal o se canceló antes de que se iniciara el Map Run.	El Map Run redriven utiliza la entrada del bucket de Amazon S3.
La entrada se pasó usando ItemReader. El Map Run produjo un error tras iniciar o intentar iniciar ejecuciones de flujos de trabajo secundarios.	La ejecución del Map Run redriven utiliza la misma entrada proporcionada en el intento de ejecución original.

Permiso de IAM para utilizar redrive con un Map Run

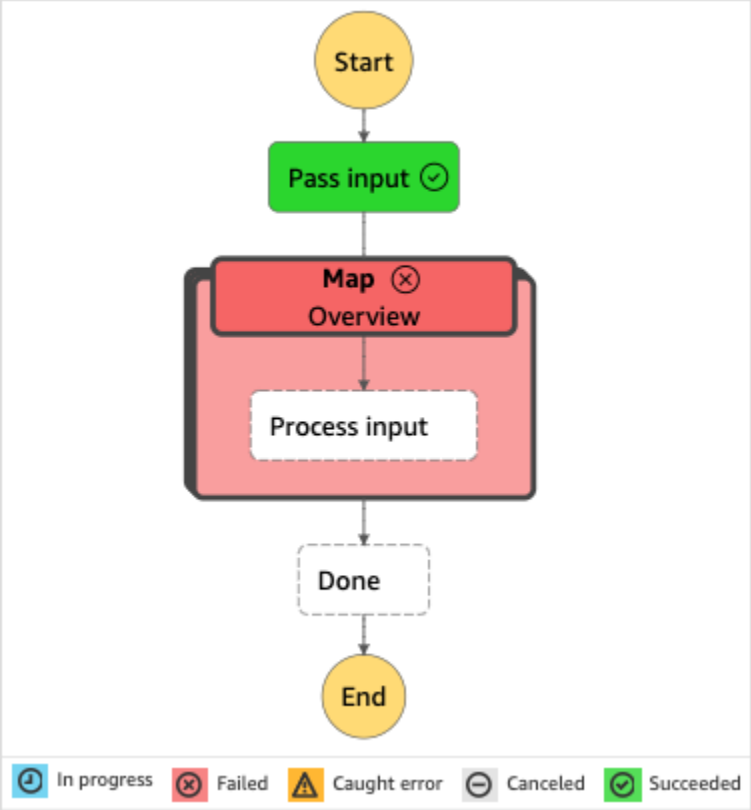
Step Functions necesita el permiso adecuado para utilizar redrive con un Map Run. El siguiente ejemplo de política de IAM concede el privilegio mínimo necesario a la máquina estatal para utilizar redriving con un Map Run. Recuerde reemplazar el texto en *cursiva* por la información específica del recurso.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

Redriving de Map Run en la consola

La siguiente imagen muestra el gráfico de ejecución de una máquina de estado que contiene un estado Map Distributed. Esta ejecución falló porque se produjo un error en el Map Run. Para utilizar redrive con el Map Run, debe utilizar redrive con el flujo de trabajo principal.



Para utilizar redrive con un Map Run desde la consola

1. Abra la consola de Step Functions y, a continuación, elija una máquina de estado existente que contenga un estado Map Distributed cuya ejecución no se haya realizado correctamente.

2. En la página de detalles de la máquina de estado, en Ejecuciones, elija una instancia de ejecución fallida de esta máquina de estado.
3. Elija Redrive.
4. En el cuadro de diálogo Redrive, elija Redrive ejecución.

> **Tip**
>
> También puede utilizar redrive con un Map Run desde la página Detalles de ejecución o Detalles del Map Run.
> Si se encuentra en la página de Detalles de la ejecución, lleve a cabo una de las siguientes acciones para utilizar redrive con la ejecución:
>
> - Elija Recuperar y, a continuación, seleccione una opción Redrive desde error.
> - Elija Acciones y, a continuación, seleccione Redrive.
>
> Si se encuentra en la página Detalles de Map Run, elija Recuperar y, a continuación, seleccione Redrive desde error.

   Observe que redrive utiliza la misma definición de máquina de estado y el mismo ARN. Continúa la ejecución desde el paso que produjo un error en el intento de ejecución original. En este ejemplo, se trata del paso del estado Map Distributed denominado Mapa y el paso de Entrada del proceso dentro de él. Tras reiniciar las ejecuciones fallidas de flujos de trabajo secundarios del Map Run, redrive continuará con la ejecución del paso Listo.
5. En la página Detalles de ejecución, seleccione Map Run para ver los detalles de la ejecución del Map Run redriven.

   En esta página, puede ver los resultados de la ejecución redriven. Por ejemplo, en la sección Resumen de ejecución de Map Run, puede ver el Recuento de Redrive, que representa el número de veces que se ha utilizado redriven con el Map Run. En la sección Eventos, puede ver los eventos de ejecución relacionados con redrive adjuntos a los eventos del intento de ejecución original. Por ejemplo, el evento `MapRunRedriven`.

Después de ejecutar redriven un mapa, puedes examinar sus redrive detalles en la consola o mediante las acciones GetExecutionHistoryy de la DescribeExecutionAPI. Para obtener más

información acerca de cómo examinar una ejecución redriven, consulte Examen de las ejecuciones redriven.

Redriving de Map Run mediante la API

Puede utilizar redrive con un Map Run apto mediante la API RedriveExecution del flujo de trabajo principal. Esta API reinicia las ejecuciones fallidas de flujos de trabajo secundarios en un Map Run.

En AWS Command Line Interface (AWS CLI), ejecute el siguiente comando para utilizar redrive con una máquina de estados con error. Recuerde reemplazar el texto en *cursiva* por la información específica del recurso.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Tras ejecutar redriven un mapa, puedes examinar sus redrive detalles en la consola o mediante la acción de la DescribeMapRunAPI. Para examinar los redrive detalles de las ejecuciones de flujos de trabajo estándar en una ejecución de mapas, puedes usar la acción GetExecutionHistoryo de la DescribeExecutionAPI. Para obtener más información acerca de cómo examinar una ejecución redriven, consulte the section called “Examen de las ejecuciones redriven”.

Puede examinar los detalles de redrive de las ejecuciones de flujos de trabajo rápidos en un Map Run en la consola de Step Functions si ha activado el registro en el flujo de trabajo principal. Para obtener más información, consulte Registro mediante CloudWatch Logs.

Control de errores en Step Functions

Todos los estados, excepto los estados `Pass` y `Wait`, pueden encontrar errores de tiempo de ejecución. Los errores pueden ocurrir por varios motivos, como los siguientes:

- Problemas con la definición de la máquina de estado (por ejemplo, un estado `Choice` no tiene ninguna regla)
- Errores de tareas (por ejemplo, una excepción de una función de AWS Lambda)
- Problemas transitorios (por ejemplo, eventos relacionados con las particiones de red)

De forma predeterminada, cuando un estado registra un error, AWS Step Functions hace que la ejecución falle por completo.

> **Tip**
>
> Para implementar un ejemplo de flujo de trabajo que incluya control de errores en la Cuenta de AWS, consulte el Módulo 8: Control de errores de El workshop de AWS Step Functions.

Temas

Nombres de error

Step Functions identifica los errores en Amazon States Language mediante cadenas que distinguen entre mayúsculas y minúsculas, denominadas nombres de error. Amazon States Language define un conjunto de cadenas integradas que designan errores conocidos; todas comienzan por el prefijo `States.`.

`States.ALL`

Comodín que coincide con cualquier nombre de error conocido.

> **Note**
>
> Este tipo de error no puede detectar el tipo de error de terminal de `States.DataLimitExceeded` ni los tipos de error de tiempo de ejecución. Para obtener más información sobre estos tipos de error, consulte `States.DataLimitExceeded` y `States.Runtime`.

`States.DataLimitExceeded`

Step Functions informa de una excepción `States.DataLimitExceeded` en las condiciones siguientes:

- Cuando la salida de un conector supera la cuota de tamaño de carga útil.
- Cuando la salida de un estado supera la cuota de tamaño de la carga útil.

- Cuando, después del procesamiento de `Parameters`, la entrada de un estado es mayor que la cuota de tamaño de la carga útil.

Para obtener más información sobre cuotas, consulte Cuotas.

> **Note**
>
> Se trata de un error de terminal que el tipo de error `States.ALL` no puede detectar.

`States.ExceedToleratedFailureThreshold`

Se produjo un error en un estado `Map` porque el número de elementos con errores superó el umbral especificado en la definición de la máquina de estado. Para obtener más información, consulte Umbral de error tolerado para el estado Distributed Map.

`States.HeartbeatTimeout`

Estado `Task` que no puede enviar un latido durante un período mayor que el valor de `HeartbeatSeconds`.

> **Note**
>
> Este error solo está disponible dentro de los campos `Catch` y `Retry`.

`States.IntrinsicFailure`

Se produjo un error al intentar invocar una función intrínseca dentro de una plantilla de carga útil.

`States.ItemReaderFailed`

Se produjo un error en un estado `Map` porque no se pudo leer la fuente del elemento especificada en el campo `ItemReader`. Para obtener más información, consulte `ItemReader`.

`States.NoChoiceMatched`

Un estado `Choice` no pudo encontrar coincidencias entre la entrada y las condiciones definidas en la regla de elección y no se especificó una transición predeterminada.

`States.ParameterPathFailure`

Se produce un error al intentar reemplazar un campo dentro del campo `Parameters` de un estado cuyo nombre termina en `.$` con una ruta.

`States.Permissions`

Un estado `Task` registró un error porque no tenía privilegios suficientes para ejecutar el código especificado.

`States.ResultPathMatchFailure`

Step Functions no pudo aplicar el campo `ResultPath` de un estado a la entrada que recibió el estado.

`States.ResultWriterFailed`

Un estado `Map` no pudo escribir los resultados en el destino especificado en el campo `ResultWriter`. Para obtener más información, consulte `ResultWriter`.

`States.Runtime`

Error de ejecución causado por una excepción que no se ha podido procesar. La causa de estos problemas suelen ser errores en el tiempo de ejecución, como intentar aplicar `InputPath` o `OutputPath` a una carga JSON nula. Un error `States.Runtime` no se puede volver a intentar y siempre provocará que se produzca un error en la ejecución. Un reintento o captura de `States.ALL` no capturará los errores de `States.Runtime`.

`States.TaskFailed`

Estado `Task` que registró un error durante la ejecución. Cuando se usa en un reintento o una captura, `States.TaskFailed` actúa como un comodín que coincide con cualquier nombre de error conocido excepto `States.Timeout`.

`States.Timeout`

Estado `Task` que se ejecuta durante más tiempo que el valor `TimeoutSeconds` o que no puede enviar un latido durante un período mayor que el valor de `HeartbeatSeconds`.

Además, si una máquina de estado funciona durante más tiempo que el valor `TimeoutSeconds` especificado, se produce un error `States.Timeout` en la ejecución.

Los estados pueden notificar errores con otros nombres. No obstante, estos nombres de error no pueden empezar por el prefijo `States.`.

Utilice la práctica recomendada de asegurarse de que el código de producción controle las excepciones del servicio de AWS Lambda (`Lambda.ServiceException` y

`Lambda.SdkClientException`). Para obtener más información, consulte Gestionar excepciones de servicio de Lambda.

> **Note**
>
> Los errores no controlados en Lambda se notifican como `Lambda.Unknown` en el resultado del error. Estos incluyen los errores de memoria insuficiente y los tiempos de espera de funciones. Puede buscar coincidencias de estos errores con `Lambda.Unknown`, `States.ALL` o `States.TaskFailed` para controlarlos. Cuando Lambda alcanza el número máximo de invocaciones, el error es. `Lambda.TooManyRequestsException` Para obtener más información acerca de errores de función de Lambda, consulte Tratamiento de errores y reintentos automáticos en la Guía para desarrolladores de AWS Lambda.

Reintento después de un error

Los estados `Task`, `Parallel` y `Map` pueden tener un campo llamado `Retry`, cuyo valor debe ser una matriz de objetos denominados reintentadores. Cada reintentador representa un número determinado de reintentos, que normalmente aumentan en intervalos de tiempo.

Cuando uno de estos estados informa de un error y hay un campo `Retry`, Step Functions escanea los reintentadores en el orden indicado en la matriz. Cuando el nombre del error aparece en el valor del campo `ErrorEquals` de un reintentador, la máquina de estado realiza reintentos tal como se define en el campo `Retry`.

Si la ejecución de redriven se vuelve a realizar con un Estado de la tarea, un Parallel o un estado Map en línea para el que haya definido reintentos, el recuento de reintentos para estos estados se restablecerá en 0 para permitir el número máximo de intentos en redrive. Para una ejecución redriven, puede realizar un seguimiento de los reintentos individuales de estos estados mediante la consola. Para obtener más información, consulte Comportamiento de reintento de las ejecuciones redriven en Redriving de ejecuciones.

Los reintentadores contienen los siguientes campos:

> **Note**
>
> Los reintentos se tratan como transiciones de estado. Para obtener información sobre cómo las transiciones de estado afectan a la facturación, consulte Precios de Step Functions.

`ErrorEquals` (Obligatorio)

Matriz no vacía de cadenas que coinciden con nombres de error. Cuando un estado registra un error, Step Functions lo analiza a través de los reintentadores. Cuando el nombre de error aparece en esta matriz, se implementa la política de reintentos que se describe en este reintentador.

`IntervalSeconds` (opcional)

Entero positivo que representa el número de segundos antes del primer reintento (el valor predeterminado es `1`). El valor máximo de `IntervalSeconds` es 99999999.

`MaxAttempts` (opcional)

Entero positivo que representa el número máximo de reintentos (el valor predeterminado es `3`). Si el error se repite más veces de las especificadas, se interrumpen los reintentos y se reanuda el control normal de errores. Un valor de `0` especifica que el error nunca se volverá a intentar. El valor máximo de `MaxAttempts` es 99999999.

`BackoffRate` (opcional)

Multiplicador según el cual aumenta el intervalo de reintento que indica `IntervalSeconds` después de cada intento de reintento. De forma predeterminada, el `BackoffRate` valor aumenta en `2.0`.

Por ejemplo, supongamos que `IntervalSeconds` es 3, `MaxAttempts` es 3 y `BackoffRate` es 2. El primer intento se realiza tres segundos después de que se produzca el error. El segundo reintento tiene lugar seis segundos después del primer intento. El tercer reintento tiene lugar 12 segundos después del segundo reintento.

`MaxDelaySeconds` (opcional)

Un entero positivo que establece el valor máximo, en segundos, hasta el que puede aumentar un intervalo de reintento. Es útil utilizar este campo con el campo `BackoffRate`. El valor que se especifica en este campo limita los tiempos de espera exponenciales resultantes del multiplicador de la tasa de regresión que se aplique a cada reintento consecutivo. Debe especificar un valor mayor que 0 y menor que 31622401 para `MaxDelaySeconds`.

Si no especifica este valor, Step Functions no limitará los tiempos de espera entre reintentos.

`JitterStrategy` (opcional)

Cadena que determina si se debe incluir o no la fluctuación en los tiempos de espera entre reintentos consecutivos. La fluctuación reduce los reintentos simultáneos al distribuirlos en

un intervalo de retardo aleatorio. Esta cadena acepta `FULL` o `NONE` como valores. El valor predeterminado es `NONE`.

Por ejemplo, supongamos que ha establecido `MaxAttempts` como 3, `IntervalSeconds` como 2 y `BackoffRate` como 2. El primer intento se realiza dos segundos después de que se produzca el error. El segundo reintento tiene lugar cuatro segundos después del primer intento y el tercer reintento, ocho segundos después del segundo intento. Si se establece `JitterStrategy` como `FULL`, el primer intervalo de reintento se distribuye aleatoriamente entre 0 y 2 segundos, el segundo intervalo de reintento se distribuye aleatoriamente entre 0 y 4 segundos y el tercer intervalo de reintento se distribuye aleatoriamente entre 0 y 8 segundos.

Ejemplos de campo de reintento

Esta sección incluye los siguientes ejemplos de código para `Retry`.

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> Para implementar un ejemplo de un flujo de trabajo de gestión de errores en la Cuenta de AWS, consulte el módulo Control de errores de El workshop de AWS Step Functions.

Ejemplo 1: Reintento con BackOffRate

En el siguiente ejemplo de `Retry` se realizan dos reintentos; el primero se realiza tras una espera de tres segundos. Según lo que especifique `BackoffRate`, Step Functions aumenta el intervalo entre cada reintento hasta alcanzar el número máximo de reintentos. En el ejemplo siguiente, el segundo reintento comienza después de esperar tres segundos tras el primer reintento.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
```

```
   "BackoffRate": 1
} ]
```

Ejemplo 2: Reintento con MaxDelaySeconds

En el siguiente ejemplo, se realizan tres reintentos y se limita el tiempo de espera resultante de `BackoffRate` a 5 segundos. El primer reintento se realiza tras esperar tres segundos. El segundo y el tercer reintento se realizan después de esperar cinco segundos después del reintento anterior, debido al límite máximo de tiempo de espera establecido en `MaxDelaySeconds`.

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

Sin `MaxDelaySeconds`, el segundo reintento tendría lugar seis segundos después del primer reintento y el tercer reintento tendría lugar 12 segundos después del segundo.

Ejemplo 3: Reintentar todos los errores excepto States.Timeout

El nombre reservado `States.ALL` que aparece en el campo `ErrorEquals` de un reintentador es un nombre comodín que coincide con cualquier nombre de error. Debe aparecer solo en la matriz `ErrorEquals` y debe aparecer en el último reintentador de la matriz `Retry`. El nombre `States.TaskFailed` también actúa como comodín y coincide con cualquier error excepto con `States.Timeout`.

En el siguiente ejemplo de un campo `Retry`, se reintentan todos los errores excepto `States.Timeout`.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
} ]
```

Ejemplo 4: Escenario de reintento complejo

Los parámetros de un reintentador se aplican a todas las visitas que se realizan a dicho reintentador durante la ejecución de un único estado.

Supongamos que tenemos el siguiente estado `Task`.

```
"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"
   } ]
}
```

Esta tarea falla cuatro veces seguidas y devuelve estos nombres de error: `ErrorA`, `ErrorB`, `ErrorC` y `ErrorB`. Como resultado, se produce lo siguiente:

- Los dos primeros errores coinciden con el primer reintentador y generan esperas de uno y dos segundos.
- El tercer error coincide con el segundo reintentador y genera una espera de cinco segundos.
- El cuarto error también coincide con el primer reintentador. Sin embargo, ya alcanzó su máximo de dos reintentos (`MaxAttempts`) para ese error en particular. Por lo tanto, ese recuperador falla y la ejecución redirige el flujo de trabajo al estado `Z` a través del campo `Catch`.

Estados alternativos

Los estados `Task`, `Map` y `Parallel` pueden tener un campo llamado `Catch`. El valor de este campo debe ser una matriz de objetos, denominados receptores.

Un receptor contiene los siguientes campos.

`ErrorEquals` (Obligatorio)

Matriz no vacía de cadenas que coinciden con nombres de error, especificados exactamente como se indicaron en el campo del mismo nombre del reintentador.

`Next` (Obligatorio)

Cadena que debe coincidir exactamente con uno de los nombres de estado de la máquina de estado.

`ResultPath` (opcional)

Ruta que determina qué entrada envía el receptor al estado especificado en el campo `Next`.

Cuando un estado registra un error y no hay un campo `Retry` o los reintentos no son capaces de resolver el problema, Step Functions lo analiza a través de los receptores en el orden en que aparecen en la matriz. Cuando el nombre de error aparece en el valor del campo `ErrorEquals` de un receptor, la máquina de estado adopta el estado especificado en el campo `Next`.

El nombre reservado `States.ALL` que aparece en el campo `ErrorEquals` de un receptor es un nombre comodín que coincide con cualquier nombre de error. Debe aparecer solo en la matriz `ErrorEquals` y debe aparecer en el último receptor de la matriz `Catch`. El nombre `States.TaskFailed` también actúa como comodín y coincide con cualquier error excepto con `States.Timeout`.

En el siguiente ejemplo, el campo `Catch` cambia a un estado llamado `RecoveryState` cuando una función de Lambda genera una excepción Java no controlada. De lo contrario, el campo cambia al estado `EndState`.

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
   "Next": "EndState"
} ]
```

> **Note**
>
> Cada receptor puede especificar varios errores que deben controlarse.

Salida de error

Cuando Step Functions adopta el estado especificado en un nombre catch, por lo general, el objeto contiene el campo `Cause`. El valor de este campo es una descripción del error en lenguaje natural. Este objeto se conoce como salida de error.

En este ejemplo, el primer receptor contiene un campo `ResultPath`. Funciona de manera parecida a un campo `ResultPath` del nivel superior del estado, lo que genera dos posibilidades:

- Toma los resultados de la ejecución de ese estado y los sobrescribe todos los datos de entrada del estado o parte de ellos.
- Toma los resultados y los agrega a los datos de entrada. En el caso de los errores controlados por un receptor, el resultado de la ejecución del estado es la salida de error.

Por tanto, en el primer receptor del ejemplo, el receptor agrega la salida de error a la entrada como un campo llamado `error-info` si no hay ningún campo con este nombre en los datos de entrada. A continuación, el receptor envía la entrada completa a `RecoveryState`. En el segundo receptor, la salida de error sobrescribe la entrada y el receptor solo envía la salida de error a `EndState`.

> **Note**
>
> Si no se especifica el campo `ResultPath`, este se establece de forma predeterminada en `$`, que selecciona y sobrescribe toda la entrada.

Cuando un estado tiene tanto campo `Retry` como `Catch`, Step Functions utiliza primero los recuperadores adecuados. Si la política de reintentos no resuelve el error, Step Functions aplica la transición del receptor correspondiente.

Provoca cargas útiles e integraciones de servicios

Un receptor devuelve una carga de cadenas como salida. Cuando trabaje con integraciones de servicios como Amazon Athena o AWS CodeBuild, puede que desee convertir la cadena `Cause` a JSON. El siguiente ejemplo de un estado `Pass` con funciones intrínsecas muestra cómo convertir una cadena `Cause` a JSON.

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
```

```
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

Ejemplos de máquina de estado que usan Retry y Catch

Las máquinas de estado que se definen en los ejemplos siguientes presuponen la existencia de dos funciones de Lambda: una que siempre registra errores y otra que espera lo suficiente para permitir que transcurra el tiempo de espera especificado en la máquina de estado.

Esta es una definición de una función de Lambda Node.js que nunca se ejecuta correctamente y devuelve el mensaje `error`. En los ejemplos de máquinas de estado siguientes, esta función de Lambda se llama `FailFunction`. Para obtener más información sobre la creación de una función de Lambda, consulte Paso 1: Crear una función de Lambda.

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

Esta es una definición de una función de Lambda Node.js que se suspende durante 10 segundos. En los ejemplos de máquinas de estado siguientes, esta función de Lambda se llama `sleep10`.

> **Note**
>
> Cuando cree esta función de Lambda en la consola de Lambda, no olvide cambiar el valor de Tiempo de espera en la sección Configuración avanzada de 3 segundos (valor predeterminado) a 11 segundos.

```
exports.handler = (event, context, callback) => {
    setTimeout(function(){
    }, 11000);
};
```

Control de errores con Retry

Esta máquina de estado utiliza un campo `Retry` para recuperar una función que no se ejecuta correctamente y que devuelve el nombre de error `HandledError`. La función se reintenta dos veces, con un retroceso exponencial entre los reintentos.

```
{
   "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
      }
   }
}
```

Esta variante utiliza el código de error predefinido `States.TaskFailed`, que coincide con cualquier error que una función de Lambda pueda devolver.

```
{
   "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**
>
> Es recomendable que las tareas que hacen referencia a una función de Lambda controlen las excepciones del servicio de Lambda. Para obtener más información, consulte Gestionar excepciones de servicio de Lambda.

Control de errores con Catch

En este ejemplo se utiliza un campo `Catch`. Cuando una función de Lambda devuelve un error, se recibe el error y la máquina de estado adopta el estado `fallback`.

```
{
   "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
      }
   }
}
```

Esta variante utiliza el código de error predefinido `States.TaskFailed`, que coincide con cualquier error que una función de Lambda pueda devolver.

```
{
   "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
      }
   }
}
```

Control de tiempos de espera con Retry

Esta máquina de estados usa un campo `Retry` para reintentar un estado `Task` cuyo tiempo de espera se agota, en función del valor de tiempo de espera especificado en `TimeoutSeconds`. Step Functions vuelve a intentar la invocación de la función de Lambda en este estado `Task` dos veces, con un retroceso exponencial entre reintentos.

```
{
   "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
      }
   }
}
```

Control de tiempos de espera con Catch

En este ejemplo se utiliza un campo `Catch`. Cuando se agota un tiempo de espera, la maquina de estado adopta el estado `fallback`.

```
{
   "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**
>
> Puede conservar la entrada del estado y el error utilizando `ResultPath`. Consulte Se utiliza ResultPath para incluir tanto el error como la entrada en un `Catch`.

Invocar AWS Step Functions desde otros servicios

Puede configurar varios otros servicios para invocar máquinas de estado. Según el tipo de flujo de trabajo de la máquina de estado, puede invocar máquinas de estado de forma asíncrona o sincrónica. Para invocar máquinas de estado de forma sincrónica, utilice la llamada a la API `StartSyncExecution` o la integración de Amazon API Gateway con Express Workflows. Con la invocación asíncrona, Step Functions detiene la ejecución del flujo de trabajo hasta que se devuelve un token de tarea. No obstante, esperar a que aparezca un token de tarea hace que el flujo de trabajo sea sincrónico.

Los servicios que puede configurar para invocar Step Functions incluyen:

- AWS Lambda, mediante la llamada `StartExecution`.
- Amazon API Gateway
- Amazon EventBridge
- AWS CodePipeline
- Motor de reglas de AWS IoT
- AWS Step Functions

Las invocaciones de Step Functions se rigen por la cuota de `StartExecution`. Para obtener más información, consulte:

- Cuotas

Consistencia de lectura in Step Functions

Las actualizaciones de las máquinas de estado de AWS Step Functions presentan consistencia final. Todas las llamadas a `StartExecution` que se realicen pasados unos segundos utilizarán la definición y `roleArn` (el Nombre de recurso de Amazon para el rol de IAM). Es posible que las ejecuciones que se inicien inmediatamente después de llamar a `UpdateStateMachine` utilicen la definición de la máquina de estado y el `roleArn` anteriores.

Para obtener más información, consulte los siguientes temas:

- `UpdateStateMachine` en la Referencia de la API de AWS Step Functions
- Actualización de un flujo de trabajo en Empezar con AWS Step Functions.

Etiquetado en Step Functions

AWS Step Functions es compatible con el etiquetado de máquinas de estado (tanto estándar como rápidas) y actividades. Esto puede ayudarle a realizar un seguimiento de los costes asociados con sus recursos y administrarlos, además de proporcionar una mayor seguridad en sus políticas de AWS Identity and Access Management (IAM). El etiquetado de los recursos de Step Functions permite administrarlos con AWS Resource Groups. Para obtener más información sobre los Grupos de recursos, consulte la AWS Resource Groups Guía del usuario.

En el caso de la autorización basada en etiquetas, los recursos de ejecución de máquinas de estado, como se muestra en el siguiente ejemplo, heredan las etiquetas asociadas a una máquina de estado.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Cuando se llama a DescribeExecution u otras API en las que se especifica el ARN del recurso de ejecución, Step Functions utiliza las etiquetas asociadas a la máquina de estados para aceptar o denegar la solicitud mientras realiza la autorización basada en etiquetas. Esto ayuda a permitir o denegar el acceso a las ejecuciones de máquina de estado en el nivel de las máquinas de estado.

Para revisar las restricciones relacionadas con el etiquetado de recursos, consulte Restricciones relacionadas con el etiquetado.

Temas

- Etiquetado para asignación de costos
- Etiquetado de seguridad
- Visualización y administración de etiquetas en la consola de Step Functions
- Administre etiquetas con las acciones de la API de Step Functions

Etiquetado para asignación de costos

Para organizar e identificar los recursos de Step Functions para la asignación de costes, puede agregar etiquetas de metadatos que identifiquen el objetivo de una máquina o actividad de estado. Esto es útil especialmente cuando dispone de muchos recursos. Puede utilizar las etiquetas de asignación de costos para organizar la factura de AWS de modo que refleje su propia estructura de costos. Para ello, regístrese para obtener una factura de su cuenta de AWS que incluya los valores y claves de etiquetas. Para obtener más información, consulte Configuración de un informe de asignación de costos mensual en la Guía del usuario de AWS Billing.

Por ejemplo, podría agregar etiquetas que representen el centro de costes y el objetivo de sus recursos de Step Functions, como se indica a continuación.

Recurso	Clave	Valor
StateMachine1	Cost Center	34567
	Application	Image processing
StateMachine2	Cost Center	34567
	Application	Rekognition processing
Activity1	Cost Center	12345
	Application	Legacy database

Este plan de etiquetado le permite agrupar dos tareas relacionadas con el rendimiento de las máquinas de estado en el mismo centro de costos, mientras etiqueta una actividad no relacionada con una etiqueta de asignación de costos distinta.

Etiquetado de seguridad

IAM permite controlar el acceso a los recursos en función de las etiquetas. Para controlar el acceso según las etiquetas, proporcione información sobre las etiquetas de recurso en el elemento de condición de una política de IAM.

Por ejemplo, puede restringir el acceso a todos los recursos de Step Functions que incluyan una etiqueta con la clave `environment` y el valor `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:DeleteActivity",
```

```
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Para obtener más información, consulte Control del acceso mediante etiquetas en la Guía del usuario de IAM.

Visualización y administración de etiquetas en la consola de Step Functions

Step Functions le permite ver y administrar las etiquetas de sus máquinas de estado en la consola de Step Functions. En la página Detalles de un equipo de estado, seleccione Etiquetas. Aquí puede ver las etiquetas existentes asociadas a su máquina de estado.

> **Note**
>
> Para administrar las etiquetas de las actividades, consulte Administre etiquetas con las acciones de la API de Step Functions.

Para añadir o eliminar etiquetas asociadas a su máquina de estado, seleccione el botón Administrar etiquetas.

1. Vaya a la página de detalles de una máquina de estado.
2. Seleccione Etiquetas, junto a Ejecuciones y Definición.
3. Elija Administrar etiquetas.
   - Para modificar las etiquetas existentes, edite Clave y Valor.
   - Para eliminar etiquetas existentes, elija Quitar etiqueta.
   - Para añadir una nueva etiqueta, seleccione Agregar etiqueta y escriba una Clave y un Valor.
4. Seleccione Guardar.

Administre etiquetas con las acciones de la API de Step Functions

Para administrar etiquetas mediante la API de Step Functions, utilice las siguientes acciones de la API:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

AWS Step Functions Estudio de flujo de trabajo

Workflow Studio for AWS Step Functions es un diseñador visual de flujos de trabajo con poco código que te permite crear flujos de trabajo sin servidor mediante la organización AWS de servicios. Con su drag-and-drop función o con el editor de código integrado, puede crear y editar flujos de trabajo, controlar cómo se filtran o transforman las entradas y salidas para cada estado y configurar la gestión de errores. A medida que arrastra y suelta estados para crear el flujo de trabajo, Workflow Studio valida su trabajo y genera código automáticamente. Puede revisar el código generado o actualizar la definición de la máquina de estado en el editor de código. Cuando haya terminado, puede guardar el flujo de trabajo, ejecutarlo y, posteriormente, examinar los resultados en la consola de Step Functions. Puede añadir y modificar visualmente los flujos de trabajo para orquestar los distintos servicios de su aplicación.

Para usar Step Functions Workflow Studio Cuenta de AWS, necesitará un y credenciales que proporcionen los permisos correctos para cualquier recurso que desee usar. Para obtener más información, consulte Requisitos previos para empezar con AWS Step Functions.

> **Note**
>
> Workflow Studio no es compatible con Internet Explorer 11. Si usa Internet Explorer 11 y tiene problemas con Workflow Studio, pruebe a utilizar otro navegador.

Puede acceder a Workflow Studio desde la consola de Step Functions al crear o editar un flujo de trabajo en Step Functions. También puede acceder a Workflow Studio en AWS Application Composer. Workflow Studio en Application Composer proporciona un entorno visual de IaC que facilita la incorporación de flujos de trabajo en sus aplicaciones sin servidor creadas con herramientas de IaC, como plantillas de AWS CloudFormation. Con Workflow Studio en Application Composer puede crear flujos de trabajo mediante plantillas de AWS CloudFormation. En Application Composer puede añadir un nuevo flujo de trabajo, modificar un flujo de trabajo existente y conectar pasos individuales del flujo de trabajo a otros recursos de la aplicación. Application Composer crea y actualiza automáticamente los recursos y configuraciones de CloudFormation necesarios. Esto ayuda a crear y administrar todos los recursos utilizados en flujos de trabajo en un solo lugar. También ayuda a acelerar el proceso desde la creación de prototipos de flujos de trabajo hasta su implementación en producción.

Al utilizar Workflow Studio en Application Composer también puede conectarse directamente a su proyecto local. Esto le ayuda a trabajar en su entorno de desarrollo integrado (IDE) junto con el lienzo visual. Para obtener más información, consulte Uso de Workflow Studio en Application Composer.

Temas

Información general de la interfaz

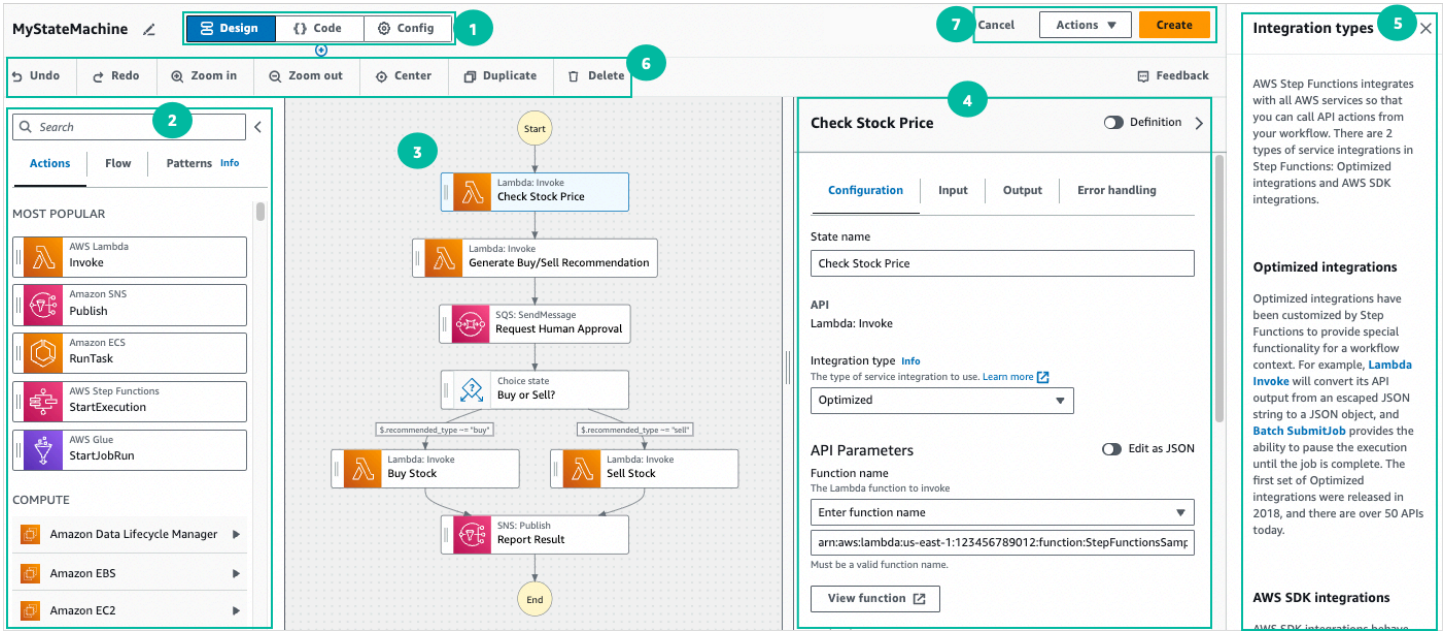
Workflow Studio for AWS Step Functions es un diseñador visual de flujos de trabajo con poco código que te permite crear flujos de trabajo sin servidor mediante la organización. Servicios de AWS Con su característica de arrastrar y soltar, Workflow Studio le facilita la creación, la edición y la visualización de los prototipos de su flujo de trabajo. Workflow Studio también ofrece un editor de código integrado para escribir y editar las definiciones del flujo de trabajo mediante Lenguaje de estados de Amazon (ASL) en la consola de Step Functions.

Para ayudarle a crear y visualizar sus flujos de trabajo, editar sus definiciones y administrar su configuración, Workflow Studio ofrece tres modos: Diseño, Código y Config. En las siguientes secciones se describen estos modos de manera detallada.

En este tema

Modo Diseño

El modo Diseño de Workflow Studio brinda una interfaz gráfica para visualizar los flujos de trabajo a medida que crea sus prototipos. En la siguiente imagen se muestran los distintos componentes disponibles en el modo Diseño.

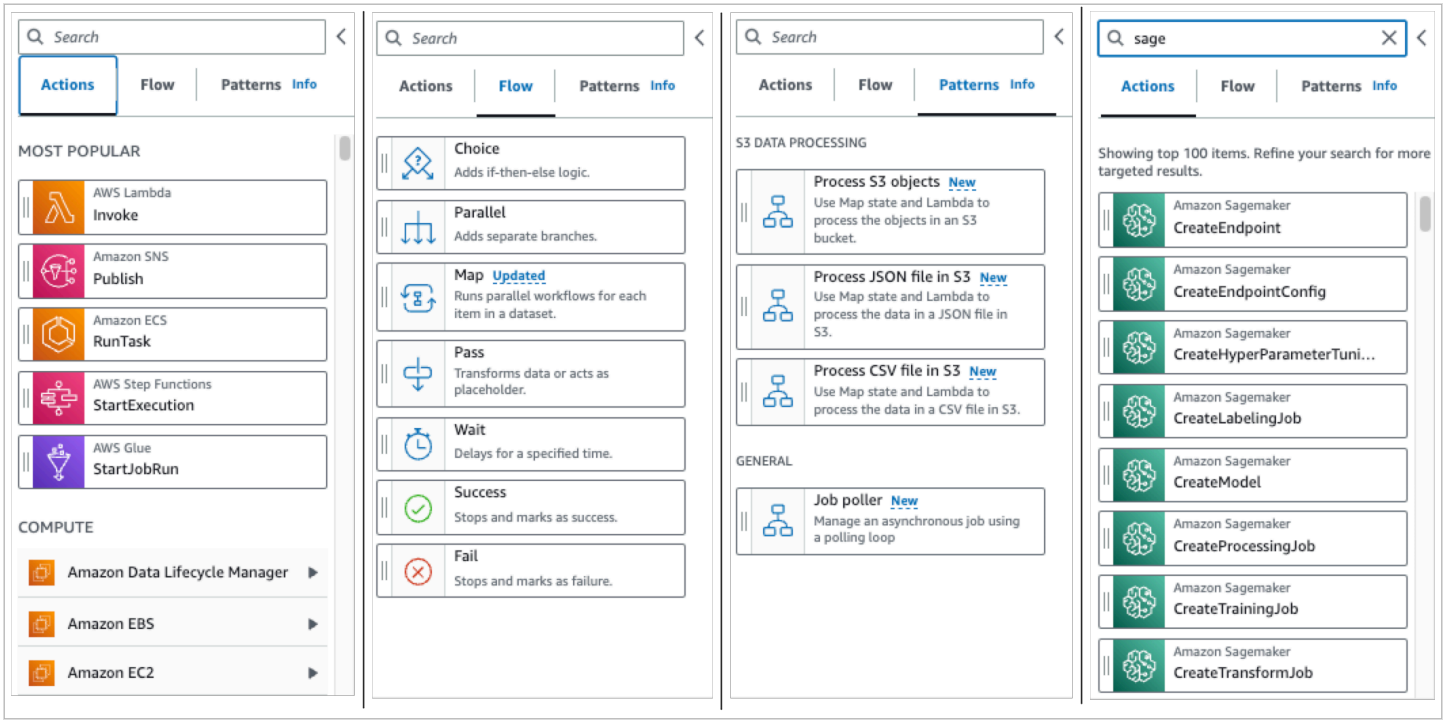


1. Botones de modo: cambie entre los modos Diseño, Código y Config de Workflow Studio con los botones de modo. No se puede cambiar de modo si el JSON de la definición de ASL del flujo de trabajo no es válido.
2. El Navegador de estados contiene las tres pestañas siguientes:
   - La pestaña Acciones proporciona una lista de AWS API que puedes arrastrar y soltar en el gráfico de flujo de trabajo del lienzo. Cada acción representa un estado Estado de la tarea.
   - La pestaña Flujo proporciona una lista de estados de flujo que puede arrastrar y soltar en el gráfico del flujo de trabajo del lienzo.
   - La pestaña Patrones proporciona varios ready-to-use componentes básicos reutilizables que puedes usar para una variedad de casos de uso. Por ejemplo, puede usar estos patrones para procesar de forma iterativa los datos de un bucket de Amazon S3.
3. El Canvas es donde usted arrastra y suelta los estados en el gráfico del flujo de trabajo, se cambia el orden de los estados y se seleccionan los estados que se van a configurar o ver.
4. El panel Inspector es el lugar donde puede ver y editar las propiedades de cualquier estado que haya seleccionado en el lienzo. Active el botón Definición para ver el código de Amazon States Language del flujo de trabajo, con el estado actualmente seleccionado resaltado.

5. Los enlaces de Información abren un panel con información contextual cuando necesita ayuda. Estos paneles también incluyen enlaces a temas relacionados en la documentación de Step Functions.
6. Barra de herramientas de diseño: contiene un conjunto de botones para realizar acciones comunes, como deshacer, eliminar y ampliar.
7. Botones de utilidad: un conjunto de botones para realizar tareas, como guardar los flujos de trabajo o exportar sus definiciones de ASL a un archivo JSON o YAML.

Navegador de estados

El navegador de estados permite seleccionar los estados para arrastrarlos y soltarlos en el gráfico del flujo de trabajo. La pestaña Acciones proporciona una lista de AWS las API y la pestaña Flujo proporciona una lista de los estados del flujo. Mientras que la pestaña Patrones proporciona varios ready-to-use componentes básicos reutilizables que puede utilizar para una variedad de casos de uso. Puede buscar todos los estados en el navegador de estados mediante el cuadro de búsqueda de la parte superior.



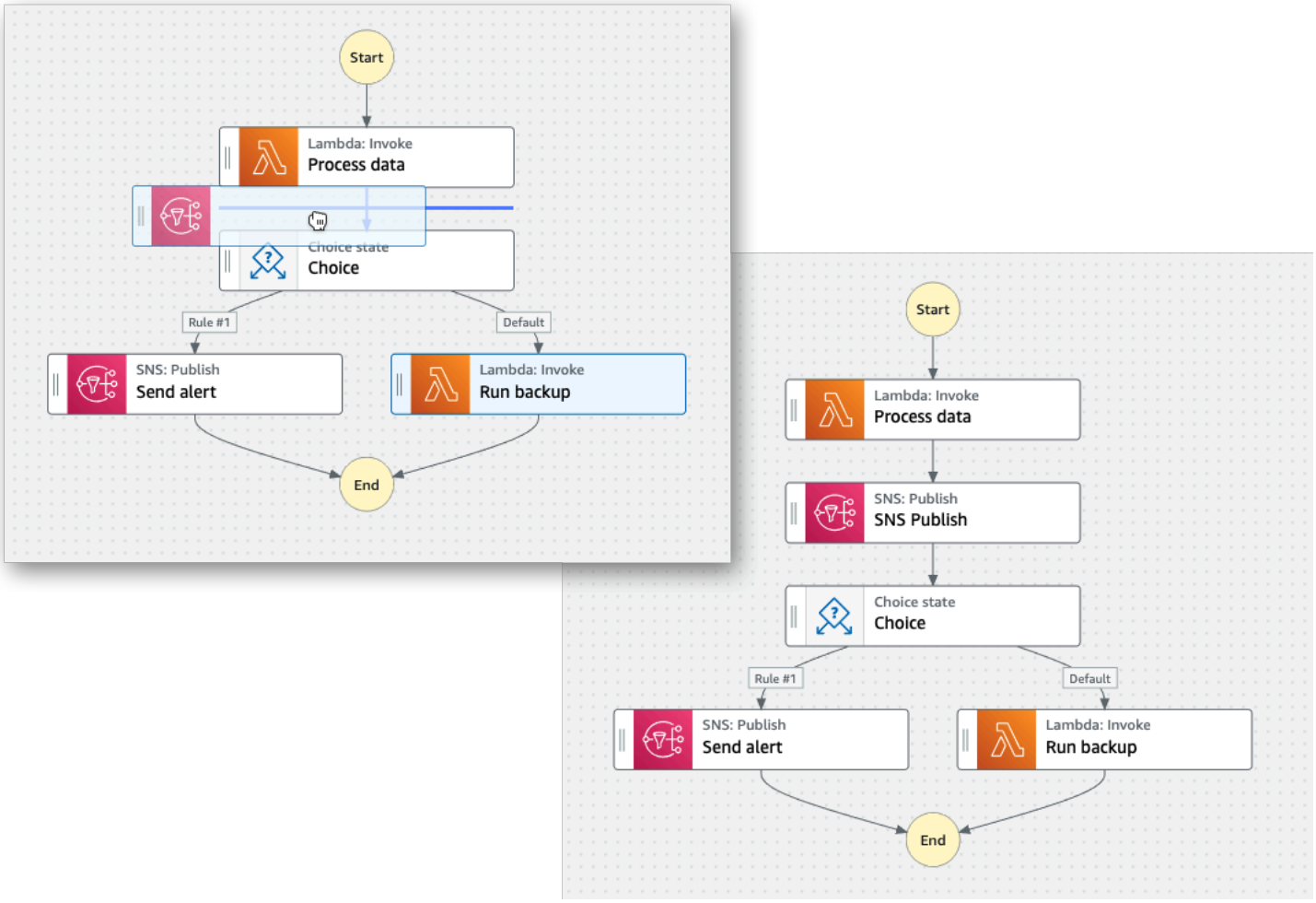
Hay siete estados de flujo que puede usar para dirigir y controlar su flujo de trabajo. Todos ellos toman la entrada del estado anterior y muchos permiten filtrar la entrada del estado anterior y la salida al estado siguiente. Los estados de flujo son:

- Choice: permite elegir entre ramificaciones de ejecución paralelas al flujo de trabajo. En la pestaña Configuración del Inspector, puede configurar reglas para determinar a qué estado pasará el flujo de trabajo.
- Parallel: permite añadir ramificaciones de ejecución paralelas al flujo de trabajo.
- Map: permite iterar pasos dinámicamente para cada elemento de una matriz de entrada. A diferencia de un estado de flujo `Parallel`, un estado `Map` ejecutará los mismos pasos para varias entradas de una matriz en la entrada de estado.
- Pass: permite pasar su entrada a su salida. (Opcional) Puede añadir datos fijos a la salida.
- Wait: permite poner el flujo de trabajo en pausa durante un tiempo determinado o hasta una hora o fecha concreta.
- Succeed: detiene el flujo de trabajo correctamente.
- Fail: detiene el flujo de trabajo con un error.

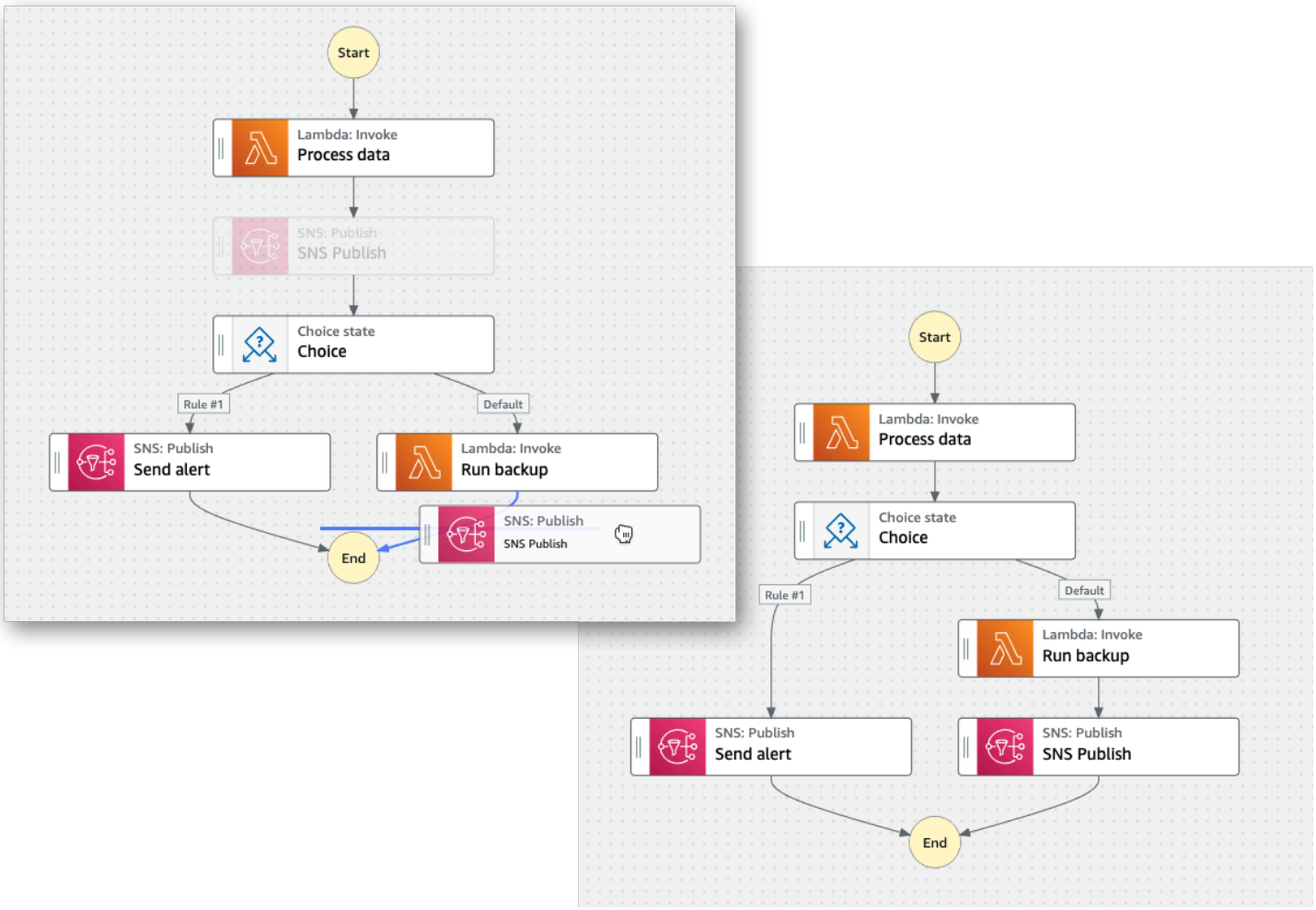
Canvas

Después de elegir un estado para añadirlo a su flujo de trabajo, arrástrelo al lienzo y suéltelo en el gráfico de flujo de trabajo. También puede arrastrar y soltar estados para moverlos a diferentes lugares del flujo de trabajo. Si su flujo de trabajo es complejo, es posible que no pueda verlo en su totalidad en el panel del lienzo. Use los controles de la parte superior del lienzo para acercar o alejar la imagen. Para ver diferentes partes del gráfico de un flujo de trabajo, puede arrastrar el gráfico del flujo de trabajo al lienzo.

Arrastre un estado de flujo de trabajo desde la pestaña Acciones o Flujo y suéltelo en su flujo de trabajo. Una línea muestra dónde se colocará en su flujo de trabajo El nuevo estado de flujo de trabajo se ha añadido a su flujo de trabajo y su código se genera automáticamente.

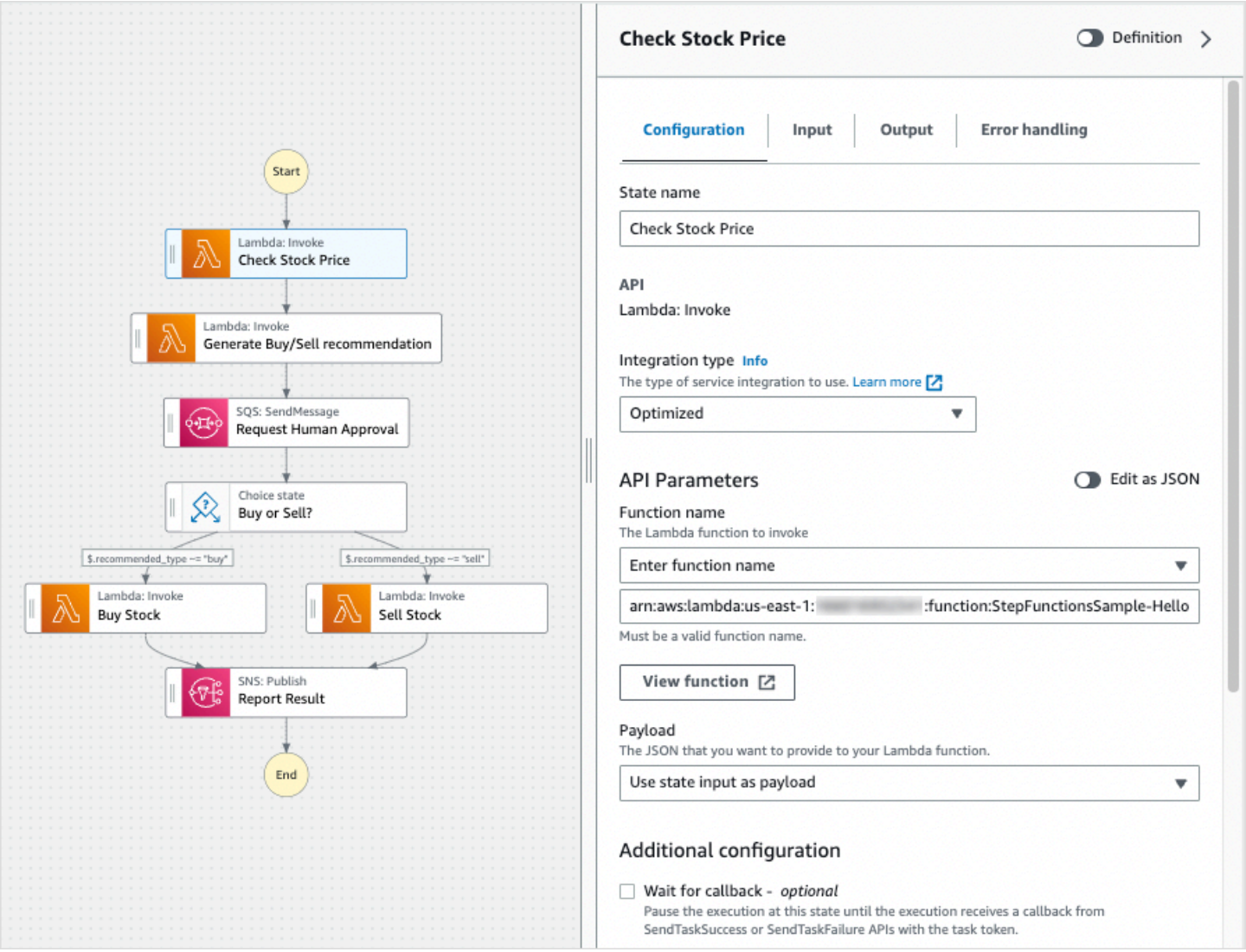


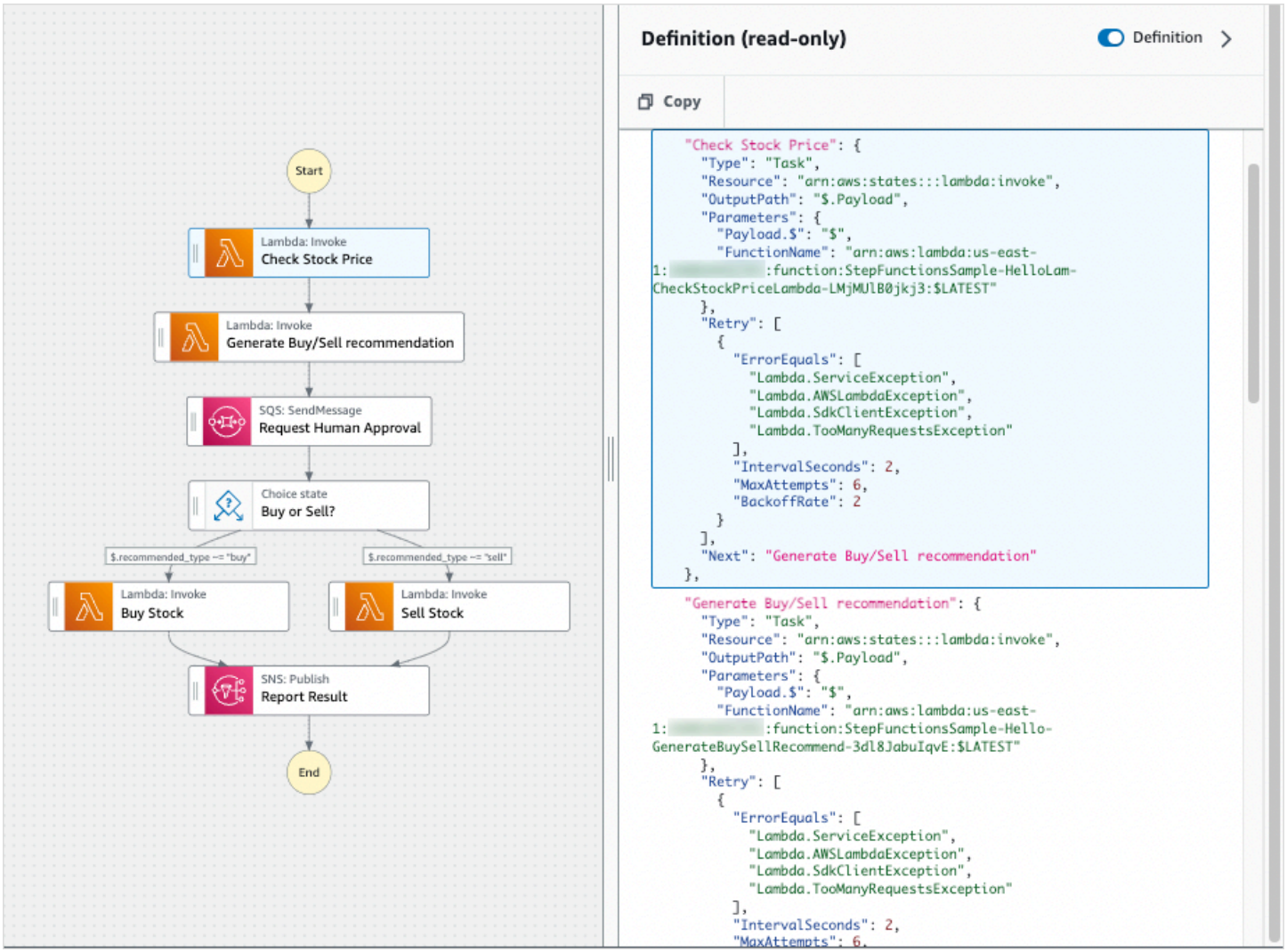
Para cambiar el orden de un estado, puede arrastrarlo a un lugar diferente de su flujo de trabajo.



Inspector

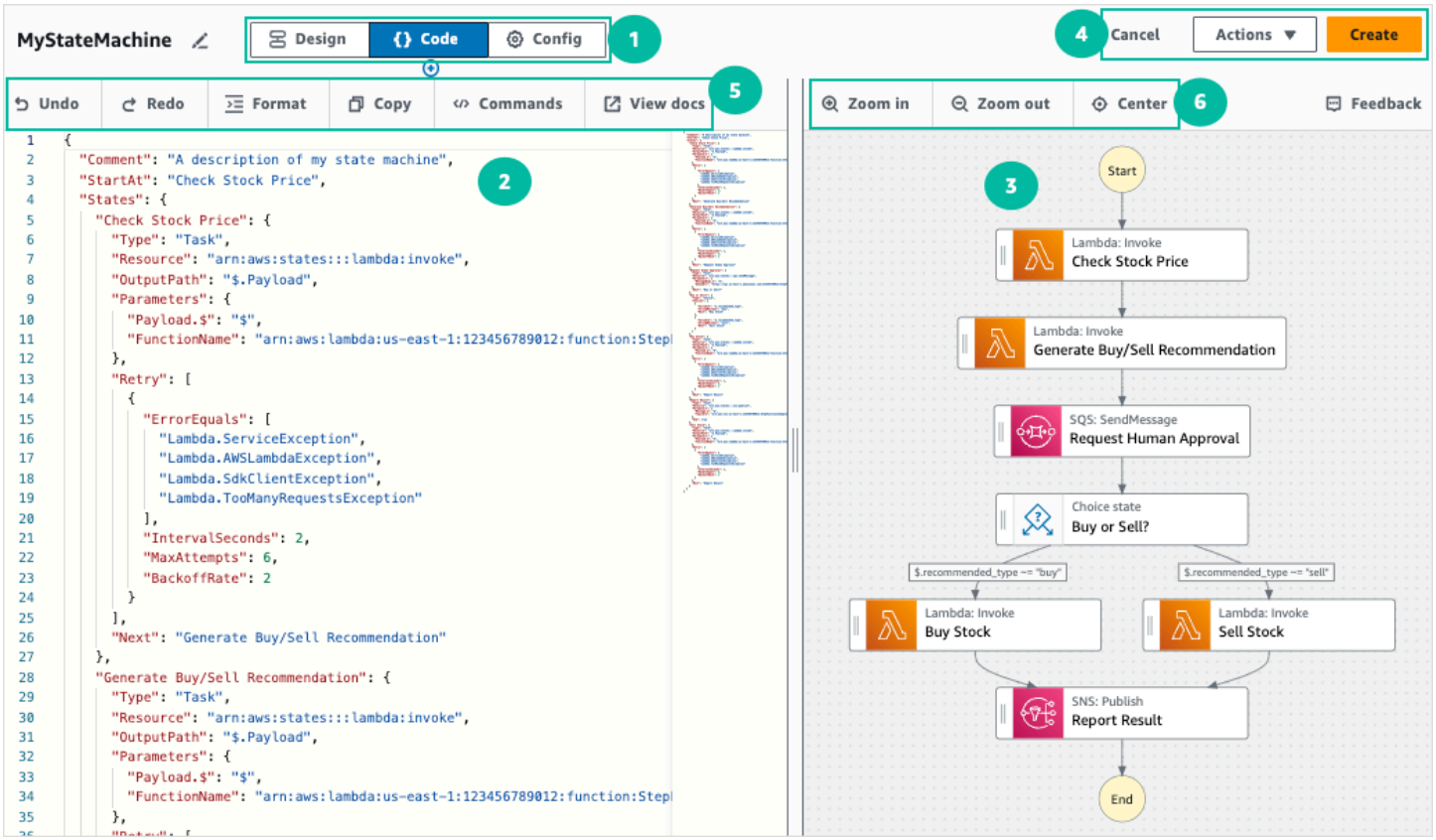
Puede configurar cualquier estado que añada a su flujo de trabajo. Seleccione el estado que desee configurar y verá sus opciones de configuración en el panel Inspector. Para ver la Definición de ASL generada automáticamente para el código del flujo de trabajo, active el botón Definición. La definición de ASL asociada al estado que ha seleccionado aparecerá resaltada.


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.



Modo Código

El modo Código de Workflow Studio ofrece un editor de código integrado para ver, escribir y editar la definición de Lenguaje de estados de Amazon (ASL) de sus flujos de trabajo en la consola de Step Functions. En la siguiente imagen se muestran los distintos componentes disponibles en el modo Código.



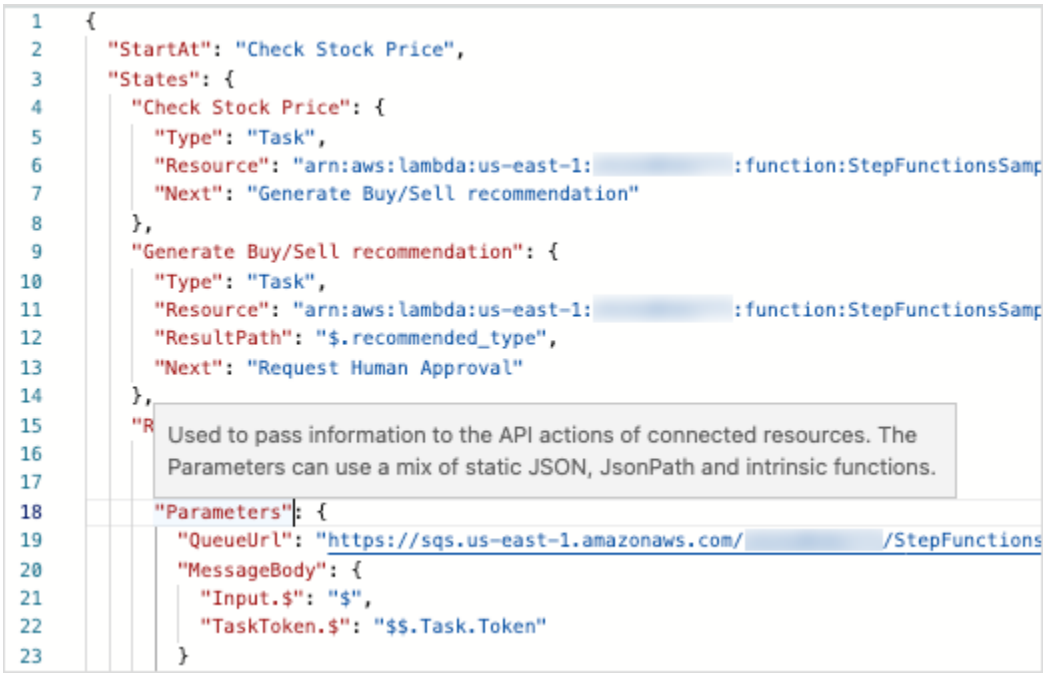
1. Botones de modo: cambie entre los modos Diseño, Código y Config de Workflow Studio con los botones de modo. No se puede cambiar de modo si el JSON de la definición de ASL del flujo de trabajo no es válido.
2. El Editor de código es el lugar donde escribe y edita la definición de ASL de sus flujos de trabajo en Workflow Studio. El editor de código también ofrece características, como el resaltado de la sintaxis y la función de autocompletar.
3. Panel de visualización de gráficos: muestra una visualización gráfica en tiempo real de su flujo de trabajo.
4. Botones de utilidad: un conjunto de botones para realizar tareas, como guardar los flujos de trabajo o exportar sus definiciones de ASL a un archivo JSON o YAML.
5. Barra de herramientas de código: contiene un conjunto de botones para realizar acciones comunes, como deshacer una acción o aplicar formato al código.
6. Barra de herramientas de gráficos: contiene un conjunto de botones para realizar acciones comunes, como acercar y alejar el gráfico del flujo de trabajo.

Editor de código

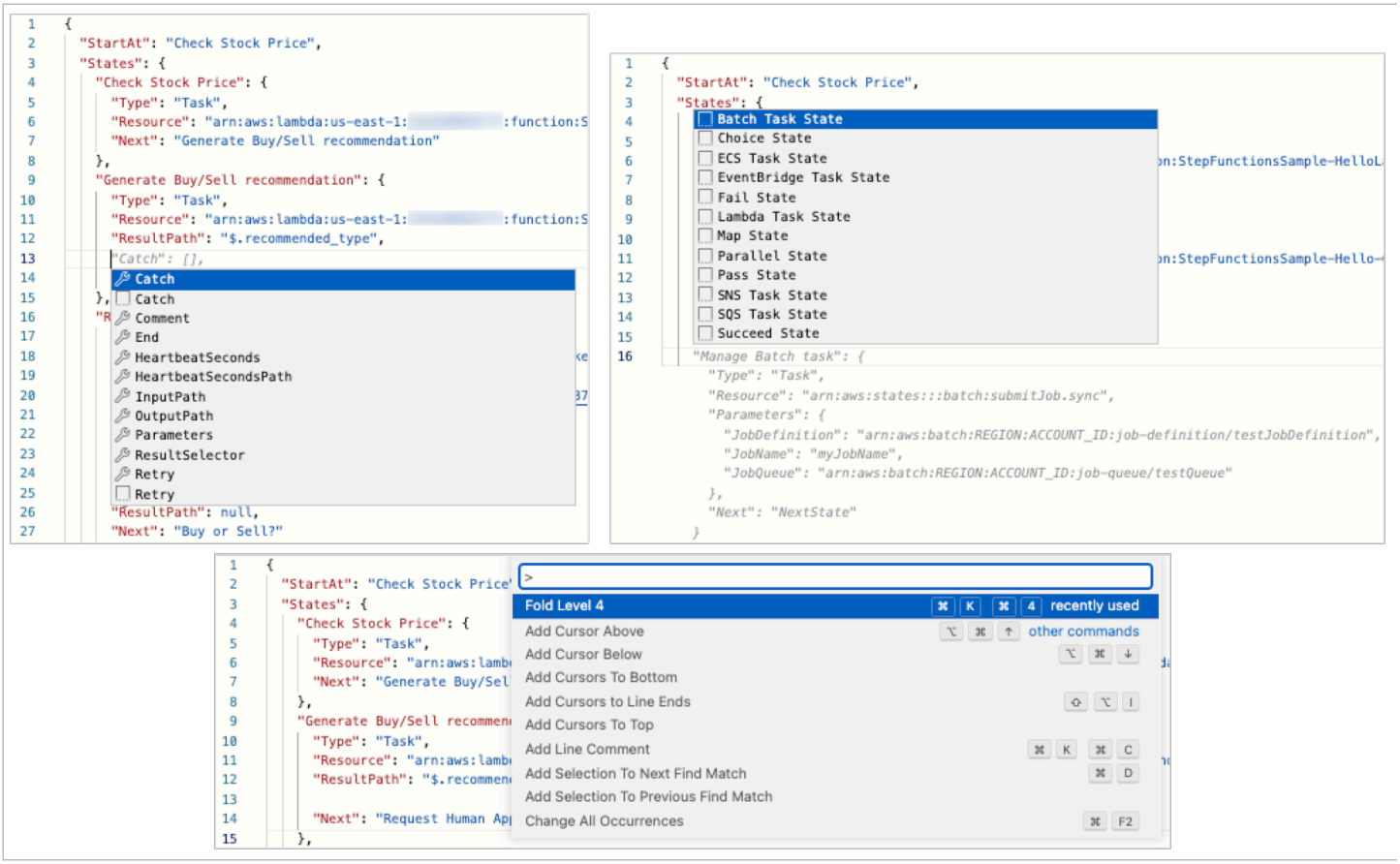
El editor de código brinda una experiencia similar a la del IDE para escribir y editar las definiciones de flujo de trabajo mediante JSON en Workflow Studio. El editor de código incluye varias características, como el resaltado de la sintaxis, las sugerencias que se completan automáticamente, la validación de la definición de ASL y la pantalla de ayuda sensible al contexto. A medida que actualiza la definición del flujo de trabajo, el Panel de visualización de gráficos representa un gráfico en tiempo real de su flujo de trabajo. También puede ver el gráfico del flujo de trabajo actualizado en el Modo Diseño.

Si se selecciona un estado en el Modo Diseño o en el panel de visualización de gráficos, la definición de ASL de ese estado aparece resaltada en el editor de código. La definición de ASL del flujo de trabajo se actualiza automáticamente si reordena, elimina o añade un estado en el modo Diseño o en el panel de visualización de gráficos.

Coloque el cursor sobre cualquier campo de la definición del flujo de trabajo para ver su ayuda sensible al contexto en forma de información sobre herramientas.



Las sugerencias que se completan automáticamente muestran fragmentos de código de los campos o estados que puede incluir en sus flujos de trabajo. Para ver una lista de campos que puede incluir en un estado específico, pulse **`Ctrl+Space`**. Para generar un fragmento de código para un nuevo estado de su flujo de trabajo, pulse **`Ctrl+Space`** después de la definición del estado actual. También puede pulsar **`F1`** para que aparezca una lista de comandos disponibles.



Panel de visualización de gráficos

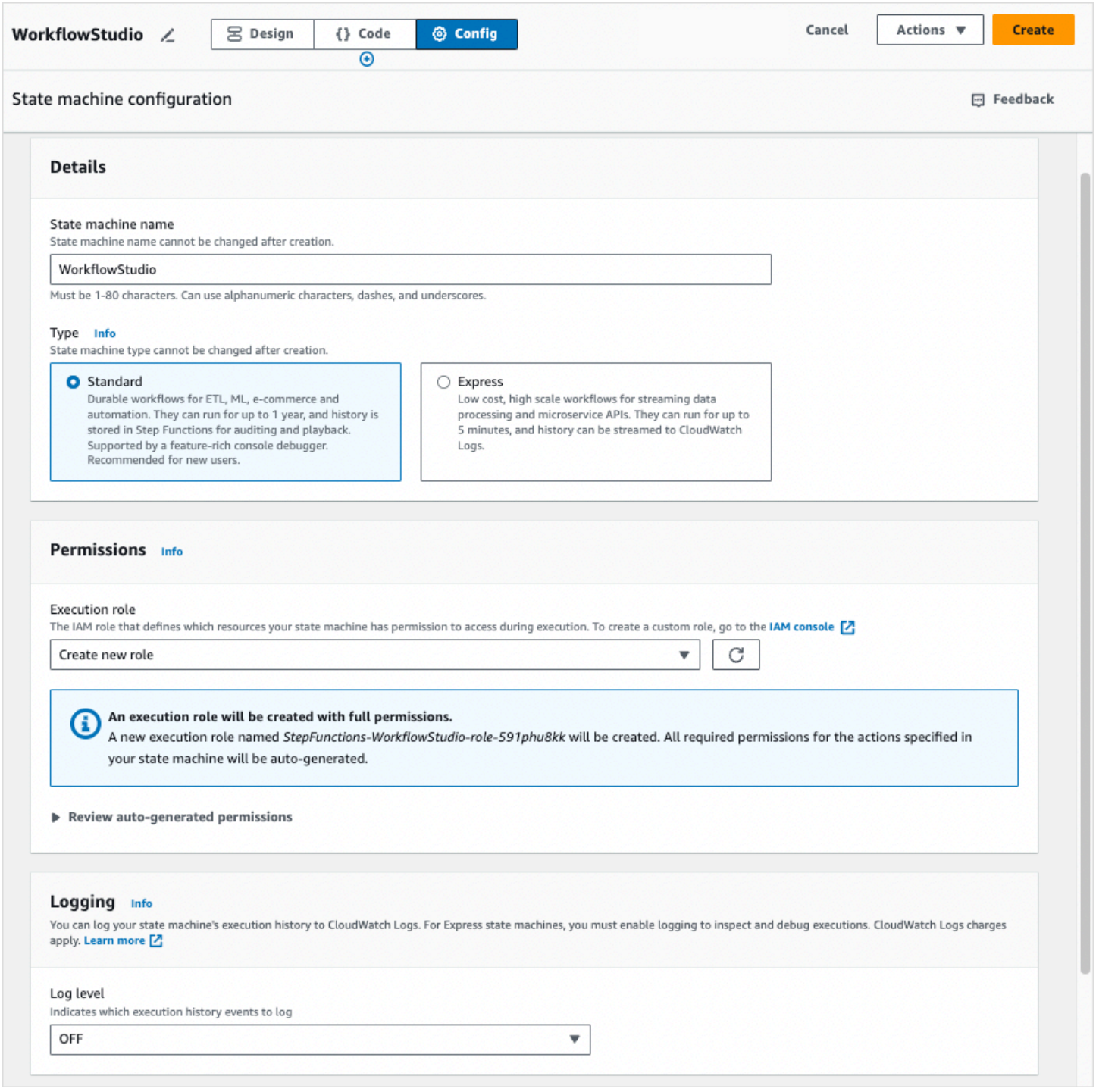
Las visualizaciones de gráficos le permiten ver cómo se ve su flujo de trabajo en formato gráfico. Al escribir las definiciones del flujo de trabajo en el Editor de código de Workflow Studio, el panel de visualización de gráficos muestra un gráfico en tiempo real del flujo de trabajo. Al reordenar, eliminar o duplicar un estado en el panel de visualización de gráficos, la definición del flujo de trabajo en el editor de código se actualiza automáticamente. Del mismo modo, a medida que actualiza las definiciones del flujo de trabajo, reordena, elimina o añade un estado en el editor de código, la visualización se actualiza automáticamente.

Si el JSON de la definición de ASL de su flujo de trabajo no es válido, el panel de visualización de gráficos detiene la representación y muestra un mensaje de estado en la parte inferior del panel.

Modo Config

El modo Config de Workflow Studio le permite administrar la configuración de sus máquinas de estados. En este modo, puede especificar detalles, como el nombre y el tipo de la máquina de estado, los permisos de IAM y la configuración de registro de la máquina de estado. Otras

configuraciones adicionales que puede especificar en este modo incluyen habilitar el AWS X-Ray rastreo y la publicación de una versión al crear la máquina de estados. Una vez creada la máquina de estado, puede editar todas las opciones de configuración de la máquina de estado, excepto el nombre y el tipo de la máquina de estado. En la siguiente imagen se muestran algunas de las configuraciones que puede especificar en el modo Config.



Administrar la configuración de la máquina de estado

Para administrar la configuración de la máquina de estado, haga lo siguiente:

1. Introduzca un nombre para la máquina de estado en el cuadro Nombre de la máquina de estado.

> **Tip**
>
> También puede seleccionar el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de la máquina de estado, especifique un nombre.

> **Important**
>
> No se puede editar el nombre de la máquina de estado una vez que esta se haya creado.

2. En Tipo, elija un tipo de máquina de estado, Estándar o Rápido. Para obtener más información acerca de los tipos de máquinas de estado, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

> **Important**
>
> No se puede editar el tipo de la máquina de estado una vez que esta se haya creado.

3. En Permisos, seleccione el rol de IAM que se utilizará como rol de ejecución para la máquina de estado.
   - Crear un nuevo rol (recomendado): si se selecciona esta opción, Step Functions crea automáticamente un rol de ejecución para las máquinas de estado con los privilegios mínimos necesarios al crear las máquinas de estado. Estas funciones de IAM generadas automáticamente son válidas para la máquina Región de AWS en la que se crea la máquina de estados.

> **Tip**
>
> Para revisar los permisos que Step Functions generará automáticamente para su máquina de estados, seleccione Revisar los permisos generados automáticamente.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

- Elegir un rol existente: cree su propio rol de IAM para la máquina de estado y, a continuación, selecciónelo entre las opciones que se muestran en Elegir un rol existente. Asegúrese de que la política del rol incluya los permisos que desea que asuma la máquina de estado.

  Para obtener información acerca de las políticas de IAM, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

- Escribir un ARN de rol: especifique el nombre de recurso de Amazon (ARN) de un rol de IAM existente que se utilizará para esta máquina de estado. Por ejemplo, `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`.

4. En Registro, defina el nivel de registro de su máquina de estado. Step Functions registra los eventos del historial de ejecución según su selección. Puede seleccionar una de las siguientes opciones:

   - TODOS: se registran todos los tipos de eventos.
   - ERROR: se registran todos los tipos de eventos de error, como TaskFailed y ExecutionFailed.
   - GRAVE: se registran todos los tipos de eventos de error grave, como ExecutionAborted y ExecutionFailed.
   - DESACTIVADO: no se registra ningún tipo de evento.

   Para obtener más información acerca de los niveles de registro, consulte Niveles de registro.

5. En Configuración adicional, defina una o varias de las siguientes configuraciones opcionales:

   - Habilitar rastreo de X-Ray: seleccione esta casilla de verificación para que Step Functions envíe registros de seguimiento a X-Ray para ejecuciones de máquinas de estado, incluso cuando no se pasa un ID de seguimiento por un servicio ascendente. Para obtener más información, consulte AWS X-Ray y Step Functions.
   - Publicar la versión al crearla: una versión es una instantánea numerada e inmutable de una máquina de estado que se puede ejecutar. Seleccione esta casilla de verificación para publicar

una versión de la máquina de estado al crear la máquina de estado. Step Functions publica la versión 1 como la primera revisión de la máquina de estado.

Para obtener más información acerca de las versiones, consulte Versiones de máquinas de estado.

- Agregar nueva etiqueta: seleccione esta casilla para agregar etiquetas a su máquina de estado. La adición de etiquetas puede ayudarle a realizar un seguimiento de los costos asociados con sus recursos y administrarlos, además de proporcionar una mayor seguridad en sus políticas de IAM. Para obtener más información acerca de las etiquetas, consulte Etiquetado en Step Functions.

6. Seleccione Crear.
7. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede elegir Ver configuración de roles para volver al modo Config.

Métodos abreviados de teclado

Workflow Studio admite los siguientes métodos abreviados del teclado:

Método abreviado de teclado	Función
Shortcuts for the Code mode	
Ctrl+space	Auto-complete suggestions
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

Método abreviado de teclado	Función
Ctrl+D	Duplicate selected state

Usar Workflow Studio

Aprenda a crear, editar y ejecutar flujos de trabajo con Step Functions Workflow Studio. Una vez que el flujo de trabajo esté listo, puede exportarlo. También puede usar Workflow Studio para crear prototipos rápidamente.

En este tema

Crear un flujo de trabajo

En Workflow Studio, puede elegir una plantilla de inicio o bien una plantilla en blanco para crear un flujo de trabajo desde cero. En el caso de las plantillas en blanco, puede usar el modo Diseño o Código para crear su flujo de trabajo.

Una plantilla de inicio es un proyecto de ready-to-run ejemplo que crea automáticamente el prototipo y la definición del flujo de trabajo y despliega todos los AWS recursos relacionados que el proyecto necesite. Cuenta de AWS Puede usar estas plantillas de inicio para implementarlas y ejecutarlas tal cual, o bien usar los prototipos de flujo de trabajo para desarrollarlas a partir de ellos. Para obtener más información acerca de las plantillas de inicio, consulte Proyectos de muestra para Step Functions.

Crear un flujo de trabajo con plantillas de inicio

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, realice una de las siguientes acciones para elegir un proyecto de muestra, por ejemplo, el proyecto de muestra Temporizador de tareas:

- Escriba **`Task Timer`** en el cuadro Buscar por palabra clave y, a continuación, seleccione Temporizador de tareas en los resultados de búsqueda que aparecen.
- Examine los proyectos de muestra que aparecen en Todos en el panel derecho y, a continuación, seleccione Temporizador de tareas.

3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.
5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo. En Modo Diseño, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. También puede cambiar al Modo Código para actualizar la definición de Lenguaje de estados de Amazon (ASL) del flujo de trabajo.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

     > **Tip**
     >
     > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

> **Note**
>
> El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

> **Important**
>
> Se aplican cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

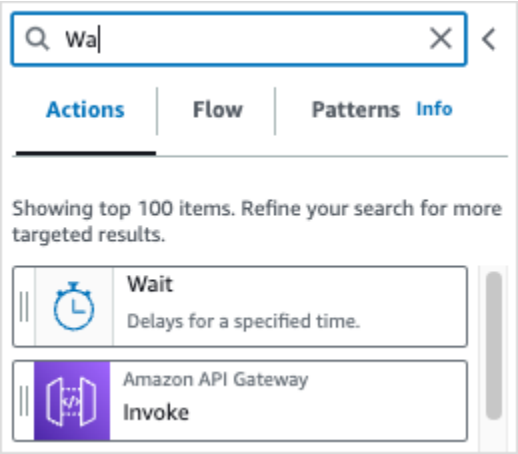
Crear un flujo de trabajo con una plantilla en blanco

1. Abra la consola de Step Functions.
2. Elija Crear máquina de estado.
3. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
4. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.

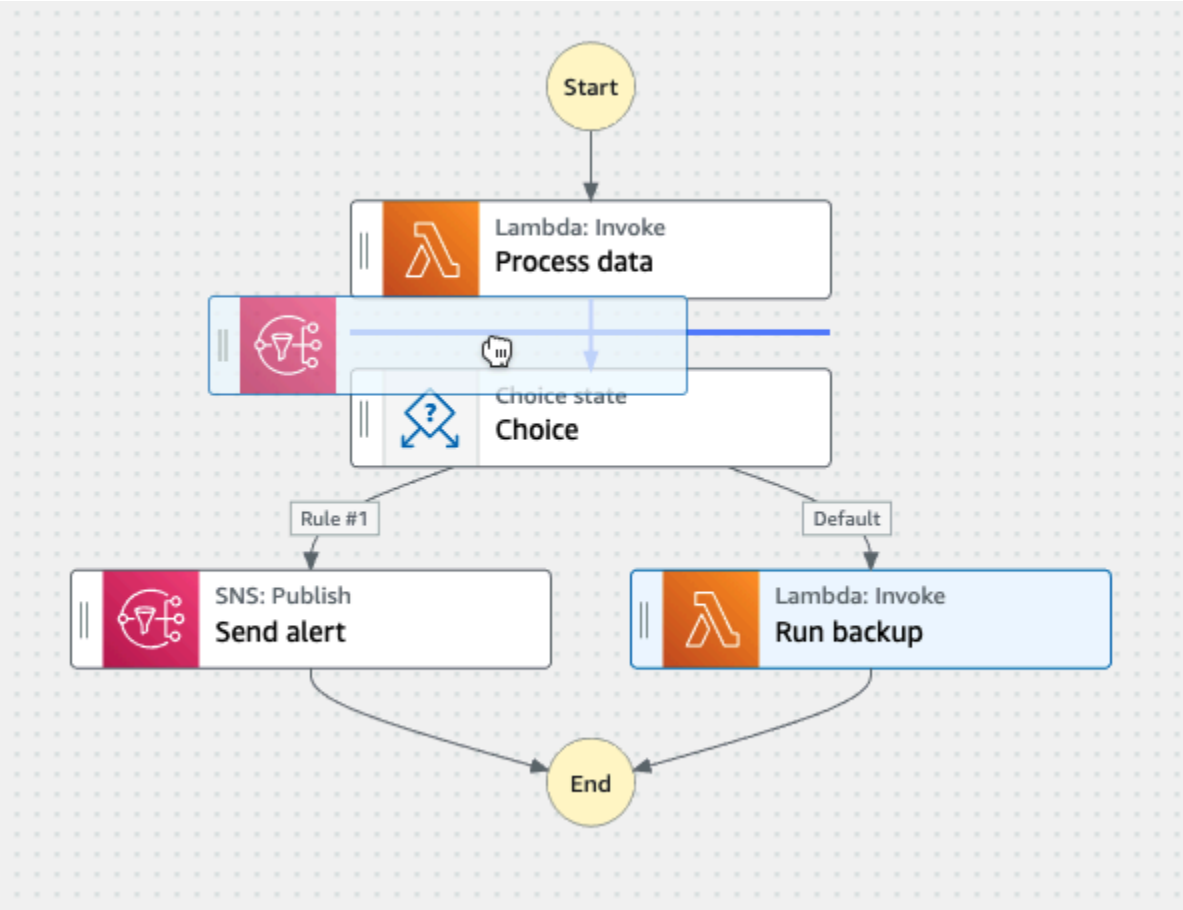
   Ahora puede empezar a diseñar su flujo de trabajo en Modo Diseño o escribir su definición de flujo de trabajo en Modo Código.
5. Seleccione Config para administrar la configuración de su flujo de trabajo en el Modo Config. Por ejemplo, proporcione un nombre para su flujo de trabajo y seleccione su tipo.

Diseñar un flujo de trabajo

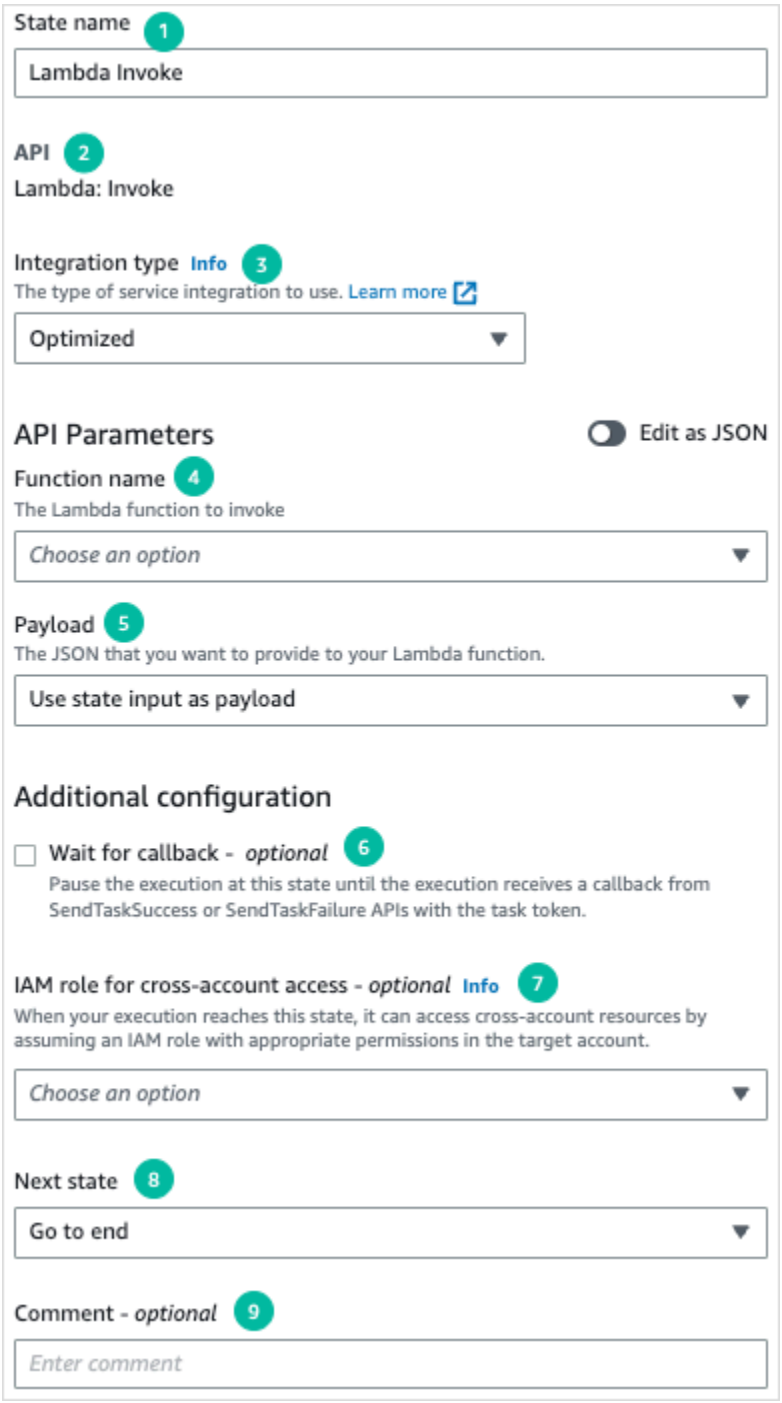
Si conoce el nombre del estado que desea añadir, utilice el cuadro de búsqueda que aparece en la parte superior del Navegador de estados para buscar ese estado en las pestañas Acciones y Flujo del Modo Diseño.



De lo contrario, seleccione un estado en el navegador de estados y arrástrelo y suéltelo en el lienzo, colocándolo donde desee en su flujo de trabajo. También puede reordenar los estados del flujo de trabajo arrastrándolos a una ubicación diferente del flujo de trabajo. A medida que arrastra un estado al lienzo, aparece una línea en el lugar donde puede colocarlo en el flujo de trabajo. Una vez que se arrastra un estado en el lienzo, su código se genera automáticamente y se añade a la definición de flujo de trabajo. Para ver la definición, active el botón Definición en el panel Inspector. Para editar la definición de su flujo de trabajo, seleccione el Modo Código que ofrezca un editor de código integrado.



Después de colocar un estado en el lienzo, puede configurarlo en el panel Inspector de la derecha. Este panel contiene las pestañas Configuración, Entrada, Salida y Control de errores para cada estado o acción de API que coloque en el lienzo. En la pestaña Configuración se configuran los estados que se incluyen en los flujos de trabajo. Por ejemplo, la pestaña Configuración para la acción de la API Lambda Invoke consta de las siguientes opciones:

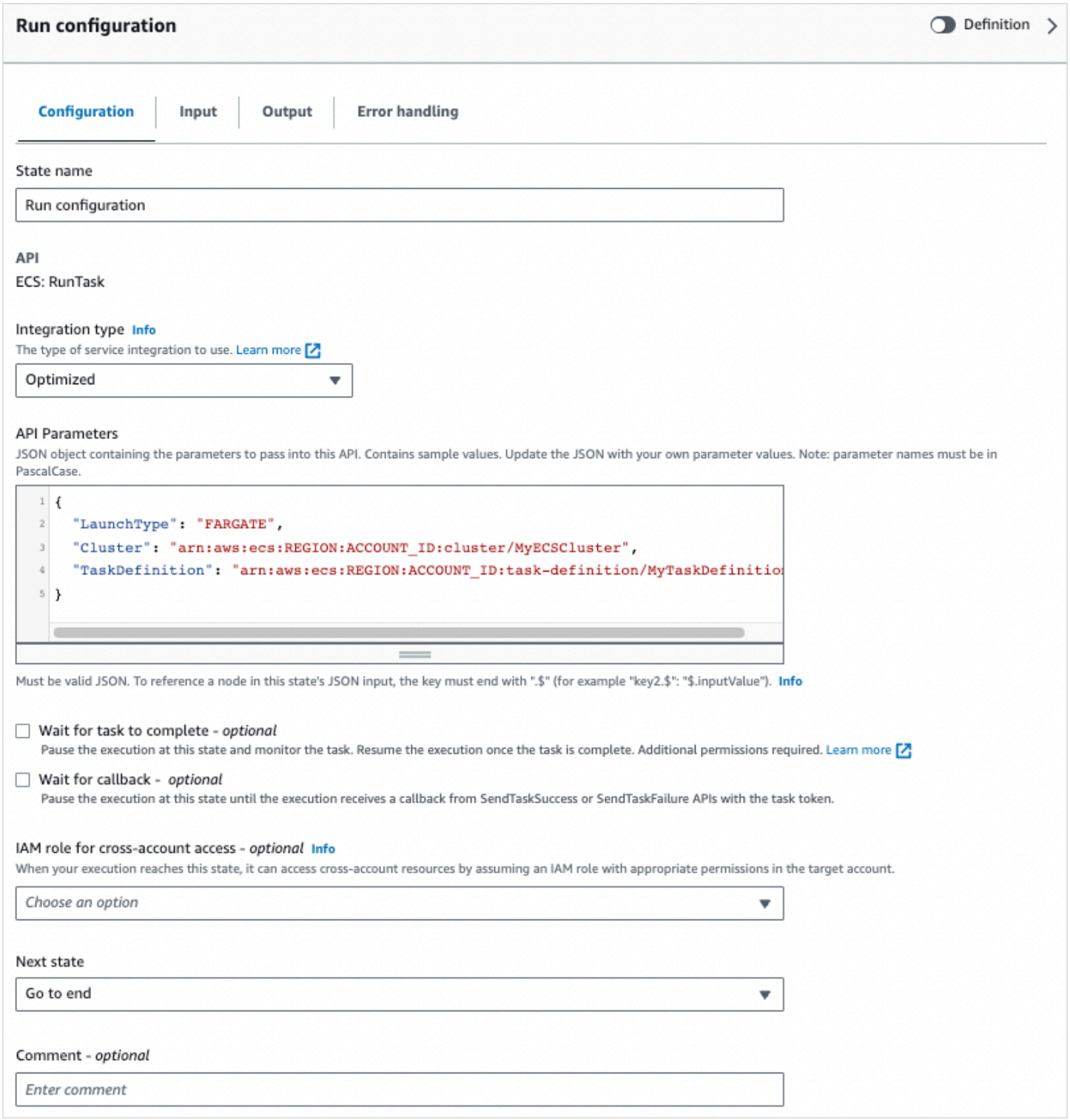


1. El nombre del estado identifica el estado. Puede usar su propio nombre o aceptar el nombre generado de manera predeterminada.
2. La API muestra la acción de API que utiliza el estado.
3. La lista desplegable Tipo de integración ofrece opciones para elegir el tipo de integraciones de servicios disponibles en Step Functions. El tipo de integración que elijas se utiliza para llamar a las acciones de la API de una parte específica Servicio de AWS de tu flujo de trabajo.

4. El nombre de la función ofrece opciones para:
   - Introducir el nombre de una función: puede introducir el nombre de la función o su ARN.
   - Obtener el nombre de una función en tiempo de ejecución a partir de la entrada de estado: puede usar esta opción para obtener dinámicamente el nombre de la función a partir de la entrada de estado en función de la ruta que especifique.
   - Seleccionar el nombre de una función: puede seleccionar directamente entre las funciones disponibles en su cuenta y región.
5. La carga le permite seleccionar entre las siguientes opciones:
   - Usar la entrada de estado como carga: puede usar esta opción para pasar la entrada del estado como la carga que se proporciona a la función de Lambda.
   - Introducir su propia carga: puede usar esta opción para construir un objeto JSON y pasarlo como carga a la función de Lambda. Este JSON puede incluir valores estáticos y valores seleccionados de la entrada de estado.
   - Sin carga: puede usar esta opción si no quiere pasar ninguna carga a la función de Lambda.
6. (Opcional) Algunos estados tendrán la opción de seleccionar Esperar a que se complete la tarea o Espere la devolución de la llamada. Cuando estén disponibles, estas opciones seleccionan uno de los siguientes patrones de integración de servicios:
   - No se ha seleccionado ninguna opción: Step Functions usará el patrón de integración Respuesta de la solicitud. Step Functions esperará una respuesta HTTP y pasará al siguiente estado. Step Functions no esperará a que se complete un trabajo. Cuando no haya opciones disponibles, el estado utilizará este patrón.
   - Esperar a que se complete la tarea: Step Functions utilizará el patrón de integración Ejecutar un trabajo (.sync).
   - Esperar la devolución de la llamada: Step Functions utilizará el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.
7. (Opcional) Para acceder a los recursos configurados de forma diferente Cuentas de AWS dentro de sus flujos de trabajo, Step Functions ofrece acceso multicuenta. El rol de IAM para el acceso entre cuentas ofrece opciones para:
   - Proporcionar el ARN del rol de IAM: permite especificar el rol de IAM que contiene los permisos de acceso a los recursos adecuados. Estos recursos están disponibles en una cuenta de destino, que es una cuenta Cuenta de AWS a la que se pueden realizar llamadas entre cuentas.
   - Obtener ARN del rol de IAM en tiempo de ejecución a partir de la entrada de estado: permite especificar una ruta de referencia a un par clave-valor existente en la entrada JSON del estado que contiene el rol de IAM.

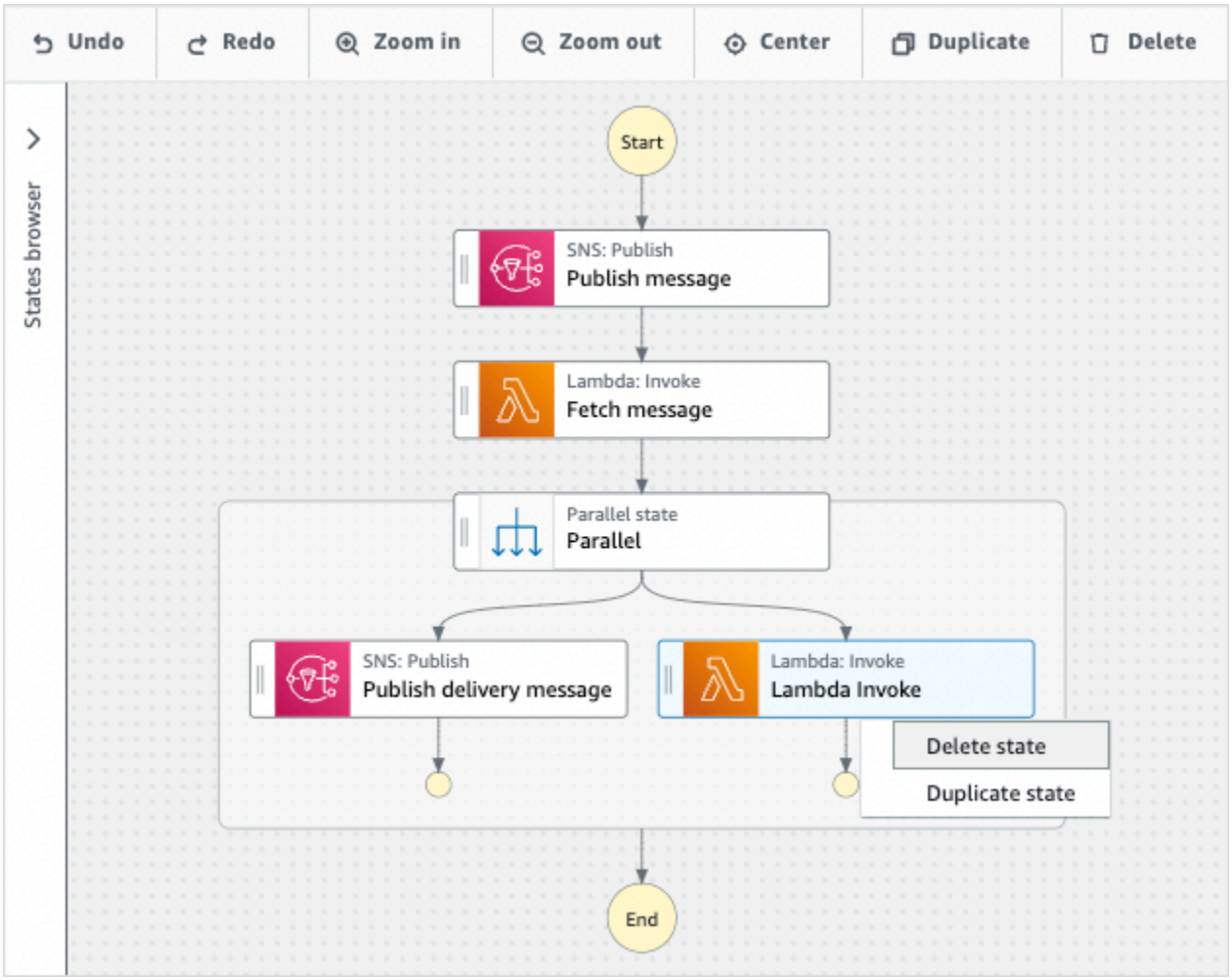
8. El estado siguiente le permite seleccionar el estado al que desea pasar a continuación.
9. (Opcional) El campo Comentario se puede utilizar para añadir su propio comentario. No afectará al flujo de trabajo, pero se puede usar para anotarlo.

Algunos estados tendrán opciones de configuración más genéricas. Por ejemplo, la configuración del estado `RunTask` de Amazon ECS contiene un campo `API Parameters` rellenado con valores de los marcadores de posición.



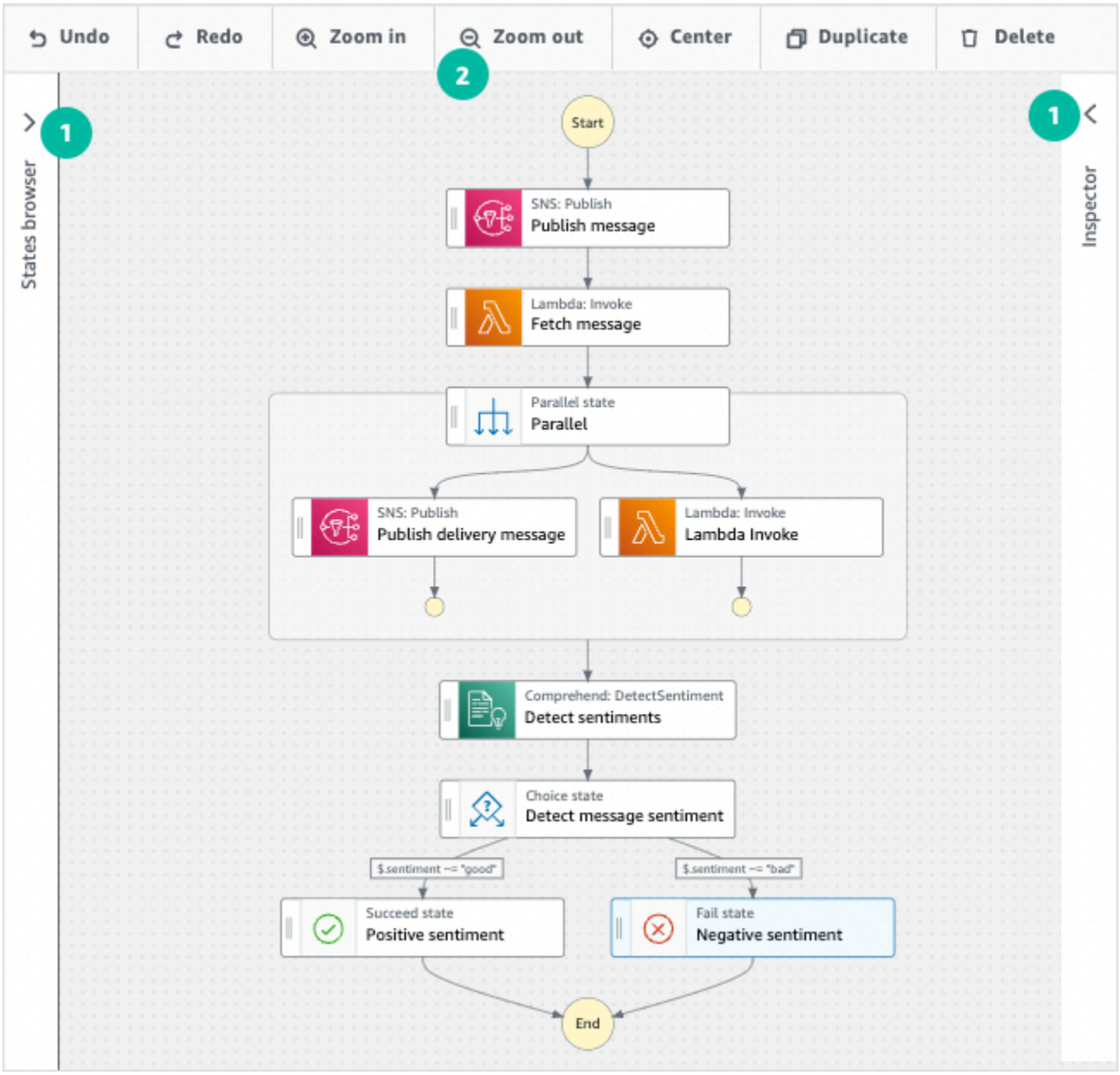
Para estos estados, puede reemplazar los valores de los marcadores de posición por configuraciones que se adapten a sus necesidades.

Para eliminar un estado, puede usar la barra de retroceso, hacer clic con el botón derecho y elegir Eliminar estado, o bien elegir Eliminar en la barra de herramientas de diseño.



A medida que el flujo de trabajo vaya creciendo, es posible que no quepa en el lienzo. Puede hacer lo siguiente:

1. Usar los controles de los paneles laterales para cambiar el tamaño de los paneles o cerrarlos.
2. Usar los controles de la barra de herramientas de diseño situados en la parte superior del Canvas para acercar o alejar el gráfico del flujo de trabajo.



Ejecutar el flujo de trabajo

Después de crear o editar el flujo de trabajo con Workflow Studio, puede ejecutarlo y ver su ejecución en la consola de Step Functions.

Para ejecutar un flujo de trabajo en Workflow Studio

1. En los modos Diseño, Código o Config, elija Ejecutar.

   Se abre el cuadro de diálogo Iniciar ejecución en una pestaña nueva.
2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.
3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

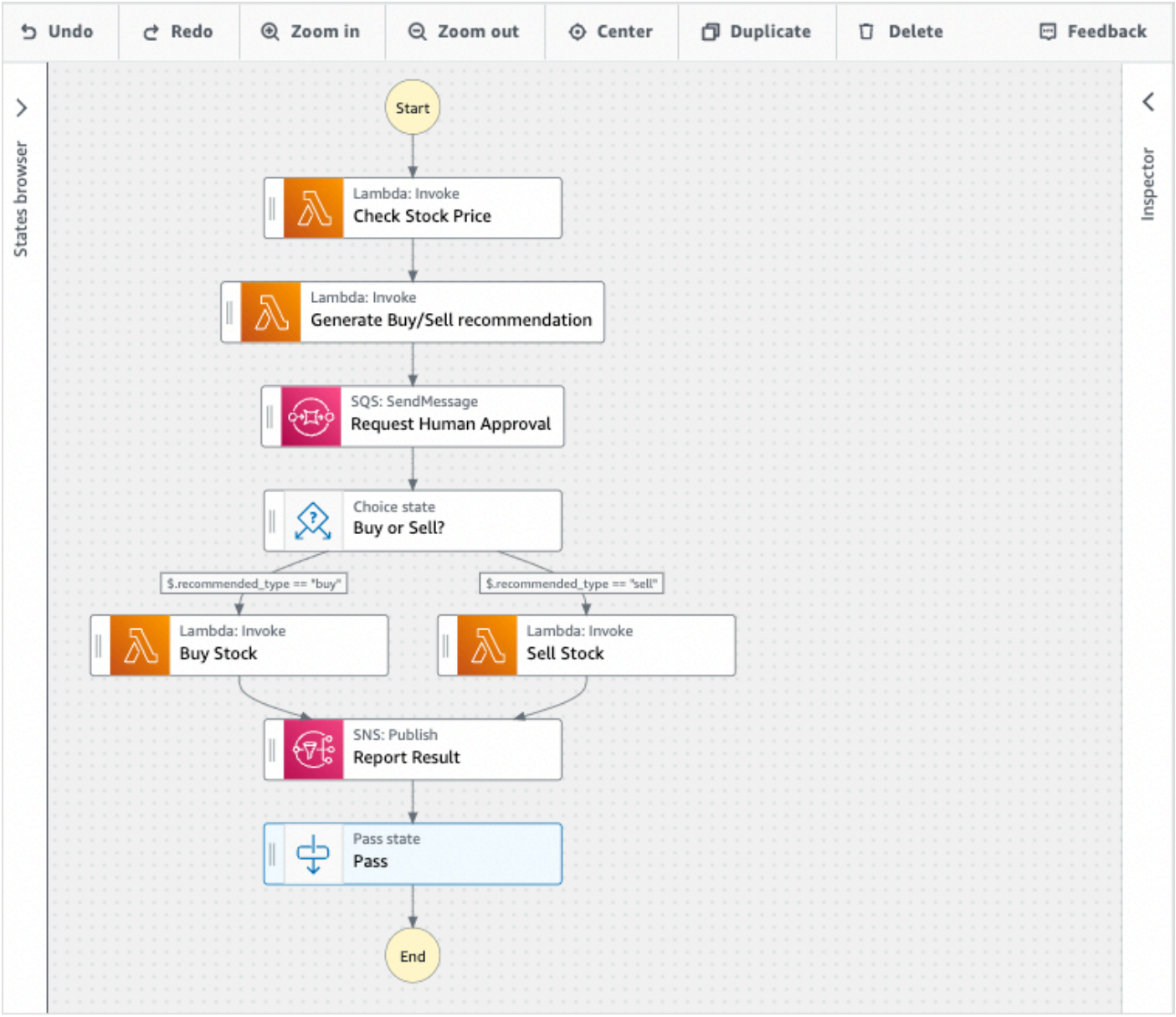
Editar el flujo de trabajo

Puede editar visualmente un flujo de trabajo existente en el Modo Diseño de Workflow Studio. También puede editar la definición del flujo de trabajo en Modo Código de Workflow Studio.

Para editar un flujo de trabajo existente:

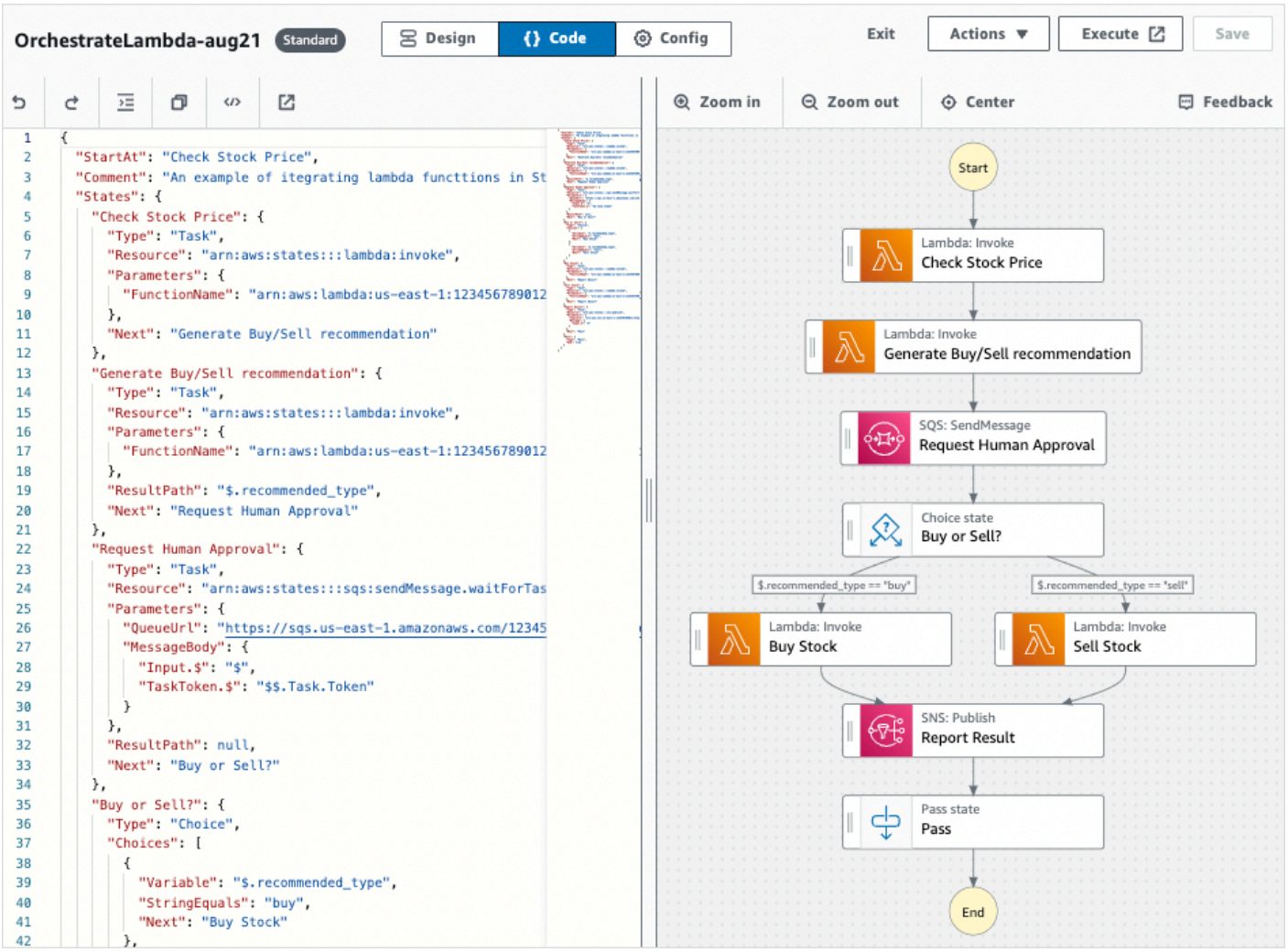
1. Abra la consola de Step Functions.
2. En la página Máquinas de estados, elija el flujo de trabajo que desee editar.
3. En la página Detalle de la máquina de estado, elija Editar.

4. El flujo de trabajo se abre en el modo Diseño de Workflow Studio. Edite el flujo de trabajo según sea necesario.



> **Note**
>
> Si se observan errores en el flujo de trabajo, deben corregirse en el modo Diseño. No se puede cambiar al modo Código o Config si hay algún error en el flujo de trabajo.

5. (Opcional) Pulse el botón Código para ver o editar la definición del flujo de trabajo en Workflow Studio.



6. Cuando haya terminado, elija Guardar para guardar el flujo de trabajo actualizado.
7. (Opcional) Para ejecutar el flujo de trabajo actualizado, seleccione Ejecutar. Se abre el cuadro de diálogo Iniciar ejecución en una pestaña nueva.

Exportar el flujo de trabajo

Puede exportar la definición de Lenguaje de estados de Amazon (ASL) de su flujo de trabajo y el gráfico de su flujo de trabajo:

1. Elija su flujo de trabajo en la consola de Step Functions.
2. En la página Detalle de la máquina de estado, elija Editar.
3. (Opcional) El flujo de trabajo se abre en el modo Diseño de Workflow Studio. Edite el flujo de trabajo en el modo Diseño o cambie al modo Código.
4. Elija el botón desplegable Acciones y luego realice una o ambas de las siguientes acciones:

- Para exportar el gráfico del flujo de trabajo a un archivo SVG o PNG, seleccione el formato que desee en Exportar gráfico.
- Para exportar la definición del flujo de trabajo como un archivo JSON o YAML, seleccione el formato que desee en Exportar definición.

Crear el prototipo del flujo de trabajo

Puede usar Workflow Studio para crear prototipos de nuevos flujos de trabajo que contengan recursos de marcadores de posición. También puede crear sus flujos de trabajo con Workflow Studio en Application Composer. Para crear un prototipo:

1. Inicie sesión en la consola de Step Functions.
2. Elija Crear máquina de estado.
3. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
4. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
5. Se abre el modo Diseño de Workflow Studio. Diseñe su flujo de trabajo en Workflow Studio. Para incluir recursos de marcadores de posición:
   a. Elija el estado para el que desea incluir un recurso de marcador de posición y, a continuación, en Configuración:
      - Para los estados Lambda Invoke, seleccione Nombre de función y luego Introducir nombre de función. También puede introducir un nombre personalizado para la función.
      - Para los estados Enviar mensaje de Amazon SQS, seleccione URL de cola y luego Introducir URL de cola. Introduzca un marcador de posición para la URL de cola.
      - Para los estados Publicar de Amazon SNS, seleccione un ARN de tema en Tema.
      - Para el resto de estados que aparecen en Acciones, puede usar la configuración predeterminada.

      > **Note**
      >
      > Si se observan errores en el flujo de trabajo, deben corregirse en el modo Diseño. No se puede cambiar al modo Código o Config si hay algún error en el flujo de trabajo.

   b. (Opcional) Para ver la definición de ASL generada automáticamente de su flujo de trabajo, seleccione Definición.

c. (Opcional) Para actualizar la definición del flujo de trabajo en Workflow Studio, pulse el botón Código.

> **Note**
>
> Si se observan errores en la definición del flujo de trabajo, deben corregirse en el modo Código. No se puede cambiar al modo Código o Config si hay algún error en la definición del flujo de trabajo.

6. (Opcional) Para editar el nombre de la máquina de estado, seleccione el icono de edición situado junto al nombre predeterminado de la máquina de estado MyStateMachiney especifique un nombre en el cuadro Nombre de la máquina de estado.

   También puede cambiar al Modo Config para editar el nombre predeterminado de la máquina de estado.

7. Especifique la configuración del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.
8. Seleccione Crear.

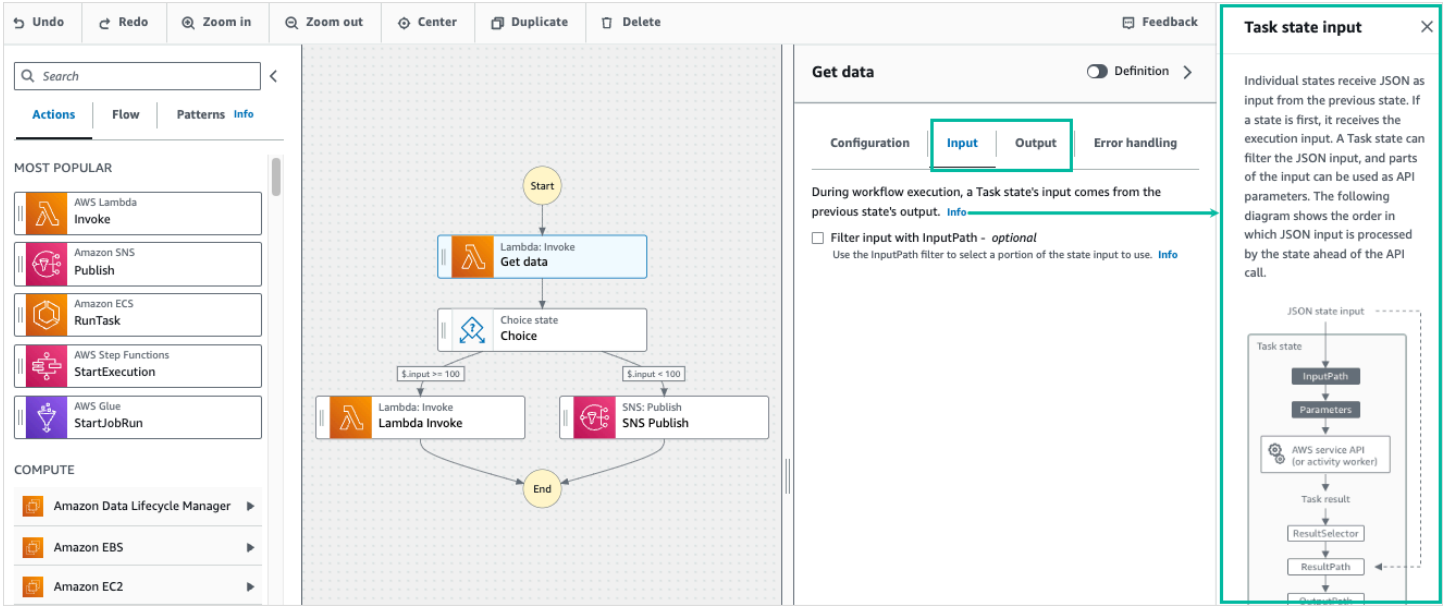
Ya ha creado un nuevo flujo de trabajo con recursos de marcadores de posición que se pueden utilizar para crear prototipos. Puede exportar la definición y gráfico de su flujo de trabajo.

- Para exportar la definición del flujo de trabajo como un archivo JSON o YAML, en el modo Diseño o Código, seleccione el botón desplegable Acciones. A continuación, en Exportar definición, seleccione el formato que desee exportar. Puede utilizar esta definición exportada como punto de partida para el desarrollo local con el AWS Toolkit for Visual Studio Code.
- Para exportar el gráfico del flujo de trabajo a un archivo SVG o PNG, en el modo Diseño o Código, seleccione el botón desplegable Acciones. A continuación, en Exportar definición, seleccione el formato que dese.

Configurar entradas y salidas para sus estados

Cada estado toma una decisión o realiza una acción en función de la información que recibe. En la mayoría de los casos, pasa la salida a otros estados. En Workflow Studio, puede configurar la forma en que un estado filtra y manipula sus datos de entrada y salida en las pestañas Entrada y Salida del

panel Inspector. Utilice los enlaces Información para acceder a la ayuda contextual al configurar las entradas y salidas.



Para obtener información detallada sobre cómo Step Functions procesada las entradas y las salidas, consulte Procesamiento de entrada y salida en Step Functions.

Configurar la entrada a un estado

Cada estado recibe la entrada del estado anterior en forma de JSON. Si desea filtrar la entrada, puede usar el filtro `InputPath` que se encuentra en la pestaña Entrada del panel Inspector. `InputPath` es una cadena que comienza por $ y que identifica un nodo JSON específico. Se denominan rutas de referencia y siguen una JsonPath sintaxis.

Para filtrar la entrada:

- Elija Filtrar la entrada con InputPath.

- Introduzca un valor válido JsonPathpara el `InputPath` filtro. Por ejemplo, **`$.data`**.

El filtro `InputPath` se añadirá a su flujo de trabajo.

Example Ejemplo 1: usar un InputPath filtro en Workflow Studio

Supongamos que la entrada a su estado incluye los siguientes datos JSON.

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Para aplicar el `InputPath` filtro, elija Filtrar la entrada con InputPath y, a continuación, introduzca una ruta de referencia adecuada. Si introduce **`$.dataset2.val1`**, el siguiente JSON se pasa como entrada al estado.

```
{"a"}
```

Una ruta de referencia también puede tener una selección de valores. Si los datos a los que hace referencia son `{ "a": [1, 2, 3, 4] }` y se aplica la ruta de referencia `$.a[0:2]` como el filtro `InputPath`, el resultado es el siguiente.

```
[ 1, 2 ]
```

Los estados de flujo Parallel, Map y Pass tienen una opción de filtrado de entrada adicional denominada `Parameters` en la pestaña Entrada. Este filtro surte efecto después del InputPath filtro y se puede usar para construir un objeto JSON personalizado compuesto por uno o más pares clave-valor. Los valores de cada par pueden ser valores estáticos, se pueden seleccionar desde la entrada o se pueden seleccionar desde un Objeto Context (Contexto) con una ruta.

> **Note**
>
> Para especificar que un parámetro utilice una ruta de referencia que apunte a un nodo JSON en la entrada, el nombre del parámetro debe terminar con `.$`.

Example Ejemplo 2: Crear una entrada JSON personalizada para el estado Parallel

Supongamos que los siguientes datos JSON son la entrada a un estado Parallel.

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Para seleccionar parte de esta entrada y transferir pares clave-valor adicionales con un valor estático, puede especificar lo siguiente en el campo Parámetros, en la pestaña Entrada del estado Parallel.

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

El resultado serán los siguientes datos JSON.

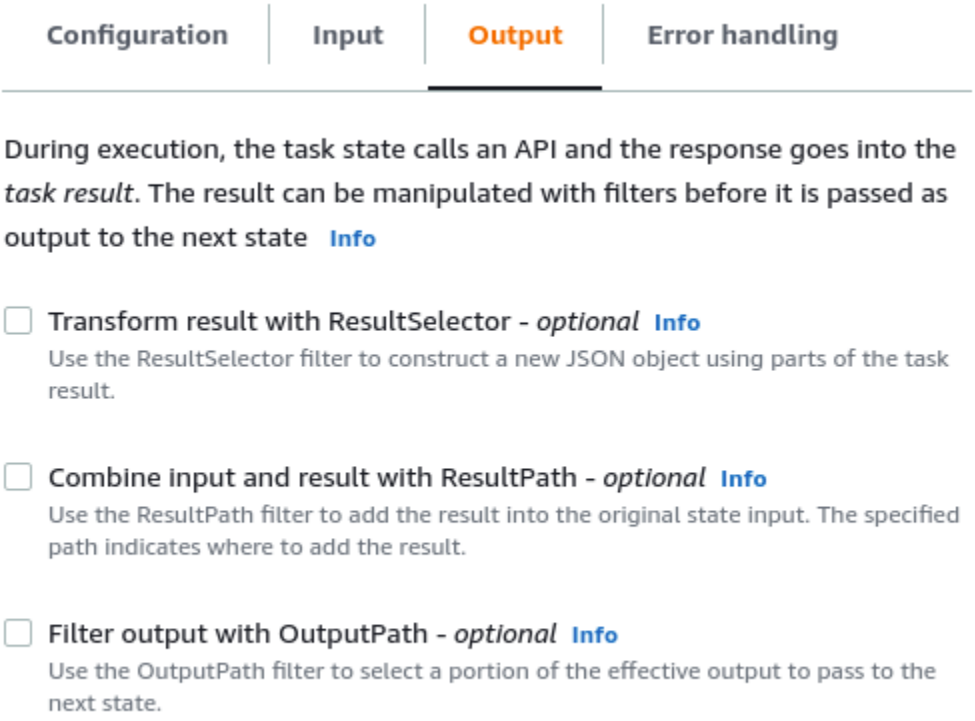
```
{
```

```
  "comment": "Selecting what I care about.",
  "MyDetails": {
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

Configurar la salida de un estado

Cada estado produce una salida JSON que se puede filtrar antes de pasar al siguiente estado. Hay varios filtros disponibles y cada uno afecta a la salida de forma diferente. Los filtros de salida disponibles para cada estado se muestran en la pestaña Salida del panel Inspector. En el caso de los estados Estado de la tarea, cualquier filtro de salida que seleccione se procesa en este orden:

1. `ResultSelector`: utilice este filtro para manipular el resultado del estado. Puede construir un nuevo objeto JSON con partes del resultado.
2. `ResultPath`: utilice este filtro para seleccionar una combinación de la entrada de estado y del resultado de tarea que transferir a la salida.
3. `OutputPath`: utilice este filtro para filtrar la salida JSON y elegir qué información del resultado se pasará al siguiente estado.



Utilice ResultSelector

`ResultSelector` es un filtro de salida opcional para los siguientes estados:

- Estados Estado de la tarea, que son todos los estados que aparecen en la pestaña Acciones del Navegador de estados.
- Estados Map, en la pestaña Flujo del navegador de estados.
- Estados Parallel, en la pestaña Flujo del navegador de estados.

`ResultSelector` se puede usar para construir un objeto JSON personalizado que consta de uno o varios pares clave-valor. Los valores de cada par pueden ser valores estáticos o seleccionarse a partir del resultado del estado con una ruta.

> **Note**
>
> Para especificar que un parámetro utilice una ruta para hacer referencia a un nodo JSON en el resultado, el nombre del parámetro debe terminar con `.$`.

Example Ejemplo de uso del ResultSelector filtro

En este ejemplo, se utiliza `ResultSelector` para manipular la respuesta de la llamada a la CreateCluster API de Amazon EMR para obtener un estado de Amazon EMR. `CreateCluster` A continuación se muestra el resultado de la llamada a la API `CreateCluster` de Amazon EMR.

```
{
  "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"
  }
}
```

Para seleccionar parte de esta información y pasar un par clave-valor adicional con un valor estático, especifique lo siguiente en el ResultSelectorcampo, en la pestaña Salida del estado.

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

El uso de `ResultSelector` produce el siguiente resultado.

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
 "ResourceType": "elasticmapreduce"
```

```
}
```

Utilice ResultPath

La salida de un estado puede ser una copia de su entrada, el resultado que produce o una combinación de su entrada y del resultado. Use `ResultPath` para controlar qué combinación de estos se pasa a la salida del estado. Para ver más casos de uso de `ResultPath`, consulte ResultPath.

`ResultPath` es un filtro de salida opcional para los siguientes estados:

- Estados Estado de la tarea, que son todos los estados que aparecen en la pestaña Acciones del navegador de estados.
- Estados Map, en la pestaña Flujo del navegador de estados.
- Estados Parallel, en la pestaña Flujo del navegador de estados.
- Estados Pass, en la pestaña Flujo del navegador de estados.

`ResultPath` se puede utilizar para añadir el resultado a la entrada de estado original. La ruta especificada indica dónde añadir el resultado.

Example Ejemplo de uso del ResultPath filtro

Supongamos que lo siguiente es la entrada a un estado Task.

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

El resultado del estado Task es el siguiente.

```
Hello, AWS Step Functions
```

Puede añadir este resultado a la entrada del estado aplicando `ResultPath` e introduciendo una ruta de referencia que indique dónde añadir el resultado, por ejemplo `$.taskresult`:

Con este `ResultPath`, lo siguiente es el JSON que se pasa como salida del estado.

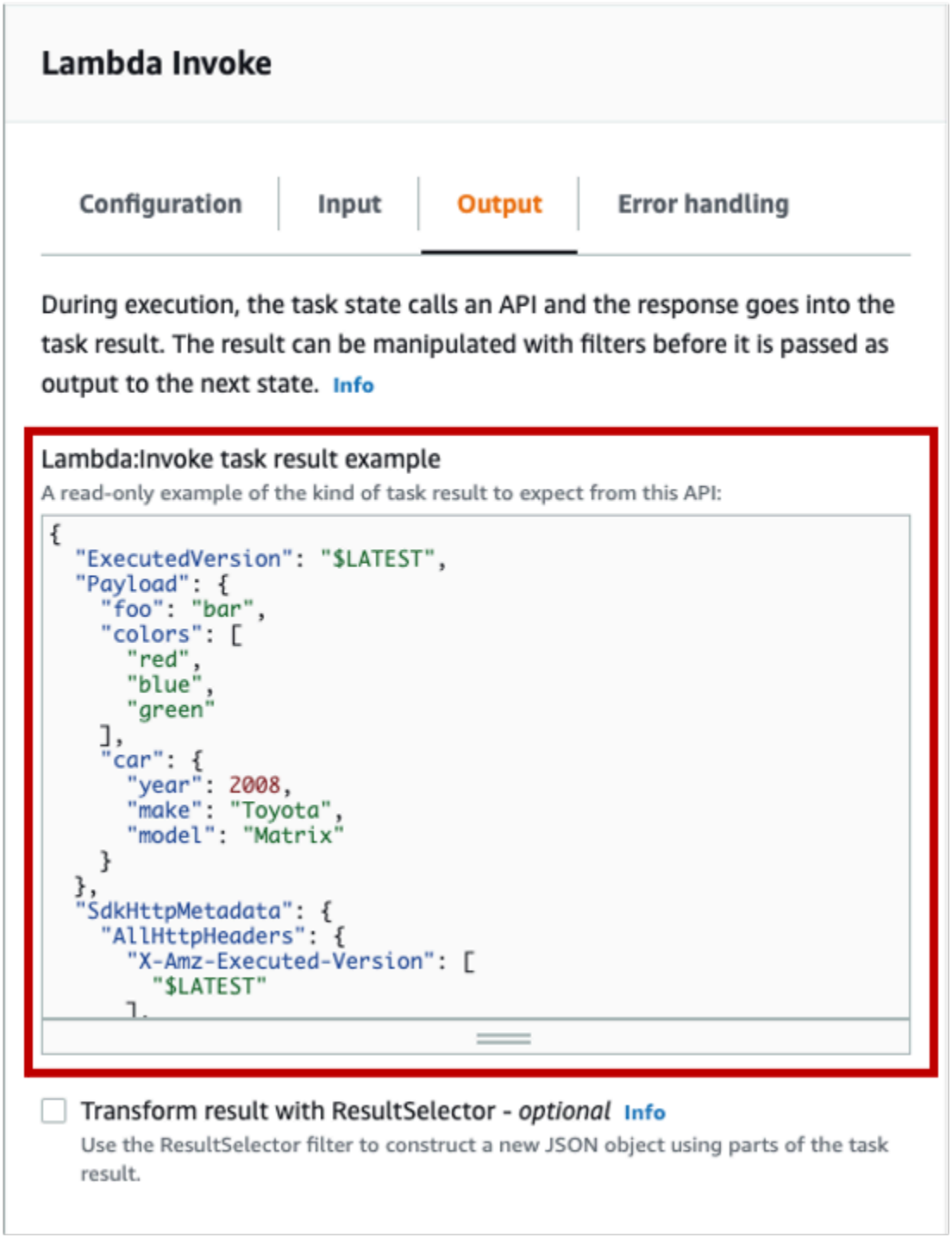
```
{
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Utilice OutputPath

El filtro `OutputPath` le permite filtrar la información no deseada y pasar solo la parte de JSON que le interese. `OutputPath` es una cadena que comienza por $ y que identifica los nodos dentro del texto JSON.

Example Ejemplo de uso del OutputPath filtro

Una llamada a la API Lambda Invoke devuelve metadatos además de la carga, que es el resultado de la función de Lambda. En la pestaña Salida del estado se muestra un ejemplo de la respuesta de esta llamada a la API.



Se puede utilizar `OutputPath` para filtrar los metadatos adicionales. De forma predeterminada, el valor del OutputPathfiltro para los estados de Lambda Invoke creados a través de Workflow Studio es. `$.Payload` Este valor predeterminado elimina los metadatos adicionales y devuelve un resultado equivalente a ejecutar directamente la función de Lambda.

El ejemplo del resultado de la tarea Lambda Invoke y el valor de `$.Payload` del filtro Salida pasan los siguientes datos JSON como salida.

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> Puesto que el filtro `OutputPath` es el último filtro de salida que surte efecto, si usa filtros de salida adicionales, como `ResultSelector` o `ResultPath`, debe modificar el valor predeterminado de `$.Payload` del filtro `OutputPath` en consecuencia.

Roles de ejecución en Workflow Studio

Cada máquina de Step Functions estado requiere un rol AWS Identity and Access Management (IAM) que le otorga permiso a la máquina de estado para realizar acciones Servicios de AWS y recursos o llamar a API de terceros. Este rol se denomina rol de ejecución. Este rol debe contener políticas de IAM para cada acción, por ejemplo, políticas que permitan a la máquina de estado invocar una función de AWS Lambda, ejecutar un trabajo de AWS Batch o llamar a la API de Stripe. Step Functions requiere que proporcione un rol de ejecución en los siguientes casos:

- Puedes crear una máquina de estados en la consola, en AWS los SDK o AWS CLI mediante la CreateStateMachineAPI.
- Puedes probar un estado en la consola, en AWS los SDK o AWS CLI mediante la TestStateAPI.

Workflow Studio cuenta con funciones que facilitan la administración de los roles de ejecución para sus flujos de trabajo.

Temas

Acerca de los roles generados automáticamente

Al crear una máquina de estado en la consola de Step Functions, Workflow Studio puede crear automáticamente un rol de ejecución que contenga las políticas de IAM necesarias. Workflow Studio analiza la definición de máquina de estado y genera políticas con los privilegios mínimos necesarios para ejecutar el flujo de trabajo.

Workflow Studio puede generar políticas de IAM para lo siguiente:

- Tareas HTTP que llaman a API de terceros.
- Estados de tareas que llaman a otros Servicios de AWS mediante integraciones optimizadas, como LambdaInvoke, DynamoDB GetItemo. AWS Glue StartJobRun
- Estados de tareas que ejecutan flujos de trabajo anidados.
- Estados Map distribuidos, incluidas políticas para iniciar las ejecuciones de flujos de trabajo secundarios, mostrar buckets de Amazon S3 y leer o escribir objetos de S3.
- Rastreo de X-Ray. Cada rol que se genera automáticamente en Workflow Studio contiene una política que otorga permisos a la máquina de estado para enviar rastros a X-Ray.
- Registro mediante CloudWatch Logs cuando está habilitado el registro en la máquina de estado.

Workflow Studio no puede generar IAM políticas para estados de tareas que llamen a otros Servicios de AWS mediante integraciones AWS del SDK.

Generación automática de roles

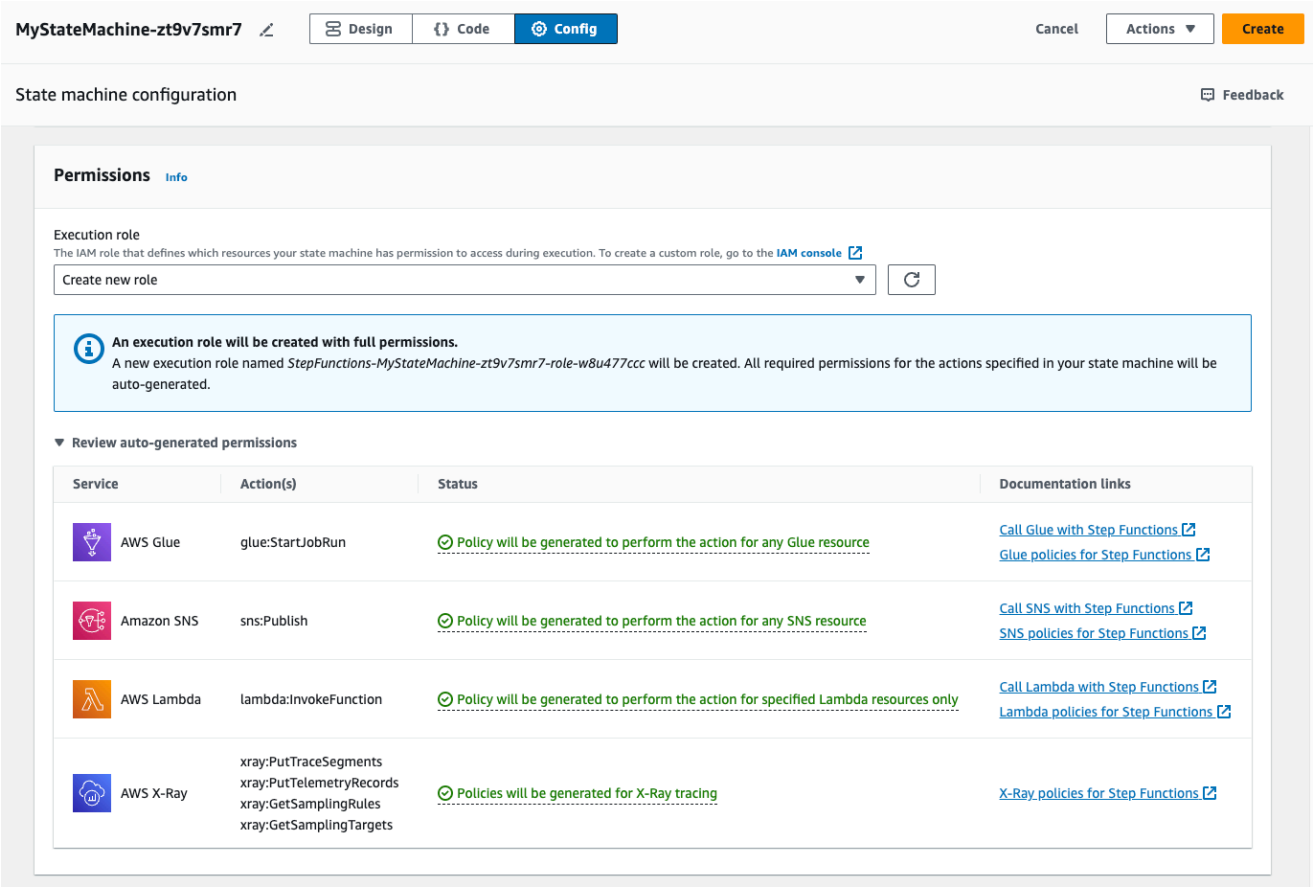
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

También puede actualizar una máquina de estado existente. Consulte el paso 4 si está actualizando una máquina de estado.

2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Seleccione la pestaña Configuración.
5. Desplácese hacia abajo hasta la sección Permisos y haga lo siguiente:

   a. En Rol de ejecución, asegúrese de mantener la selección predeterminada de Crear nuevo rol.

   Workflow Studio genera automáticamente todas las políticas de IAM necesarias para cada estado válido en la definición de máquina de estado. Muestra un aviso con el mensaje: Se creará un rol de ejecución con todos los permisos.



> **Tip**
>
> Para revisar los permisos que Workflow Studio genera automáticamente para su máquina de estado, seleccione Revisar los permisos generados automáticamente.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Si Workflow Studio no puede generar todas las políticas de IAM necesarias, mostrará un aviso con el mensaje No se pueden generar automáticamente permisos para determinadas acciones. Se creará un rol de IAM solo con permisos parciales. Para obtener información acerca de cómo añadir los permisos que faltan, consulte Resolución de problemas de generación de roles.

b. Elija Crear si va a crear una máquina de estado. De lo contrario, seleccione Save (Guardar).

c. En el cuadro de diálogo que aparece, seleccione Confirmar.

Workflow Studio guarda la máquina de estado y crea el nuevo rol de ejecución.

Resolución de problemas de generación de roles

Workflow Studio no puede generar automáticamente un rol de ejecución con todos los permisos necesarios en los siguientes casos:

- Hay errores en la máquina de estado. Asegúrese de resolver todos los errores de validación en Workflow Studio. Además, asegúrese de corregir cualquier error del lado del servidor que se produzca al guardar.
- Su máquina de estados contiene tareas, utilice integraciones de AWS SDK. En este caso, Workflow Studio no puede generar automáticamente políticas de IAM. Workflow Studio muestra un aviso con el mensaje No se pueden generar automáticamente permisos para determinadas acciones. Se creará un rol de IAM solo con permisos parciales. En la tabla Revisar permisos

generados automáticamente, elija el contenido en Estado para obtener más información sobre las políticas que faltan en su rol de ejecución. Workflow Studio puede seguir generando un rol de ejecución, pero este rol no contendrá políticas de IAM para todas las acciones. Consulte los Enlaces a la documentación para escribir sus propias políticas y añadirlas al rol una vez que se haya generado. Estos enlaces están disponibles incluso después de guardar la máquina de estado.

Rol para probar tareas HTTP en Workflow Studio

Necesita un rol de ejecución para probar el estado Task de HTTP. Si no tiene un rol con permisos suficientes, utilice una de las siguientes opciones para crearlo:

- Genera automáticamente un rol con Workflow Studio (recomendado): esta es la opción segura. Cierre el cuadro de diálogo Probar estado y siga las instrucciones que aparecen en Generación automática de roles. Para ello, primero tendrá que crear o actualizar su máquina de estado y, a continuación, volver a Workflow Studio para probar el estado.
- Use un rol con acceso de administrador: si tiene permisos para crear un rol con acceso total a todos los servicios y recursos AWS, puede usar ese rol para probar cualquier tipo de estado de su flujo de trabajo. Para ello, puede crear un rol de Step Functions servicio y añadirle la AdministratorAccess política en la IAM consola https://console.aws.amazon.com/iam/.

Rol para probar una integración de servicios optimizada en Workflow Studio

Necesita un rol de ejecución para los estados Task que llaman a integraciones de servicios optimizadas. Si no tiene un rol con permisos suficientes, utilice una de las siguientes opciones para crearlo:

- Utilice los enlaces a la documentación de Workflow Studio para escribir sus propias políticas de IAM (recomendado): esta es la opción más segura. Cierre el cuadro de diálogo Probar estado y siga las instrucciones que aparecen en Generación automática de roles. Para ello, primero tendrá que crear o actualizar su máquina de estado y, a continuación, volver a Workflow Studio para probar el estado.
- Utilice un rol con acceso de administrador: si tiene permisos para crear un rol con acceso total a todos los servicios y recursos AWS, puede usar ese rol para probar cualquier tipo de estado de su flujo de trabajo. Para ello, puede crear un rol de Step Functions servicio y añadirle la AdministratorAccess política en la IAM consola https://console.aws.amazon.com/iam/.

Función para probar la integración de un servicio de AWS SDK en Workflow Studio

Necesita un rol de ejecución para los estados Task que llaman a AWS integraciones del SDK de . Si no tiene un rol con permisos suficientes, utilice una de las siguientes opciones para crearlo:

- Utilice los enlaces a la documentación de Workflow Studio para escribir sus propias políticas de IAM (recomendado): esta es la opción más segura. Cierre el cuadro de diálogo Probar estado y siga las instrucciones que aparecen en Generación automática de roles. Para ello, primero tendrá que crear o actualizar su máquina de estado y, a continuación, volver a Workflow Studio para probar el estado. Haga lo siguiente:
  1. Cierre el cuadro de diálogo Probar estado
  2. Elija la pestaña Configuración para ver el modo Configuración.
  3. Desplácese hacia abajo hasta la sección Permisos.
  4. Workflow Studio muestra un aviso con el mensaje No se pueden generar automáticamente permisos para determinadas acciones. Se creará un rol de IAM solo con permisos parciales. Seleccione Revisar permisos generados automáticamente.
  5. La tabla Revisar permisos generados automáticamente muestra una fila que presenta la acción correspondiente al estado de tarea que desea probar. Consulte los enlaces que aparecen en Enlaces a la documentación para escribir sus propias políticas de IAM en un rol personalizado.
- Usa un rol con acceso de administrador: si tienes permisos para crear un rol con acceso total a todos los servicios y recursos AWS, puedes usar ese rol para probar cualquier tipo de estado de tu flujo de trabajo. Para ello, puede crear un rol de Step Functions servicio y añadirle la AdministratorAccess política en la IAM consola https://console.aws.amazon.com/iam/.

Rol para probar estados de flujo en Workflow Studio

Necesita un rol de ejecución para probar estados de flujo en Workflow Studio. Los estados de flujo son aquellos estados que dirigen el flujo de ejecución, como Choice, Parallel, Map, Pass, Wait, Succeed o Fail. La TestStateAPI no funciona con los estados Map o Parallel. Utilice una de las siguientes opciones para crear un rol para probar un estado de flujo:

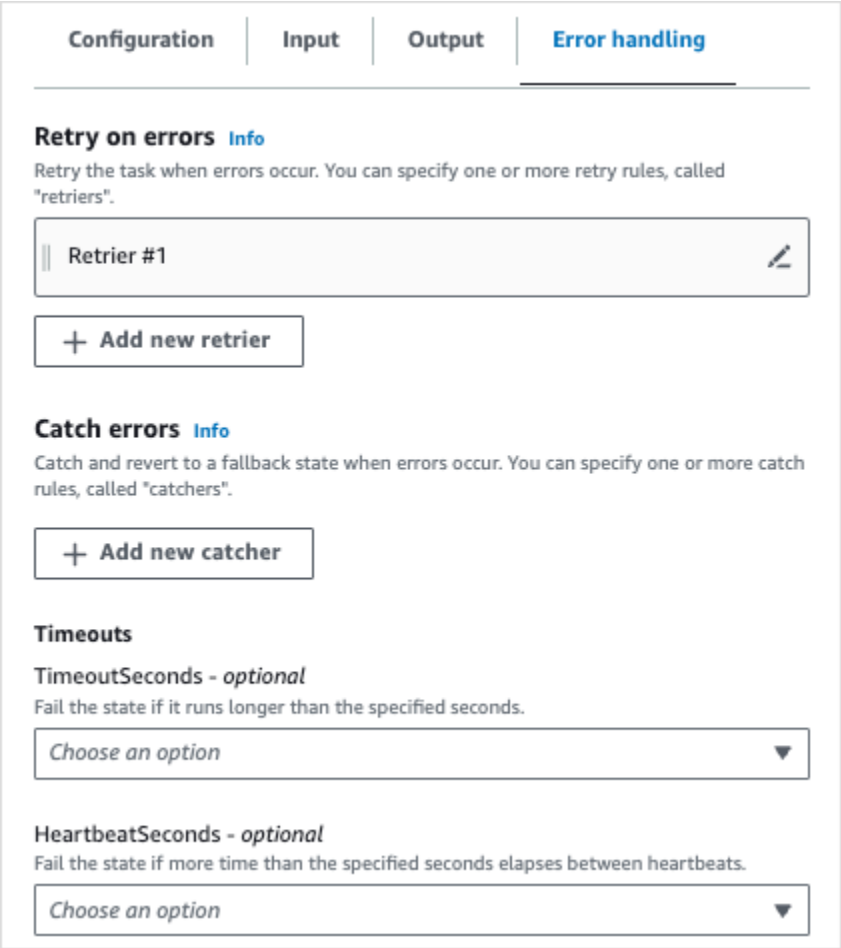
- Usa cualquier rol en tu Cuenta de AWS perfil (recomendado): los estados de flujo no requieren IAM políticas específicas, ya que no requieren AWS acciones ni recursos. Por lo tanto, puedes usar cualquier IAM rol en tu Cuenta de AWS.

1. En el cuadro de diálogo Probar estado, seleccione cualquier rol de la lista desplegable Rol de ejecución.
2. Si no aparece ningún rol en la lista desplegable, haga lo siguiente:
   a. En la consola de IAM https://console.aws.amazon.com/iam/, elija Roles.
   b. Elija un rol de la lista y copie su ARN desde la página de detalles del rol. Deberá proporcionar este ARN en el cuadro de diálogo Probar estado.
   c. En el cuadro de diálogo Probar estado, seleccione Escribir un ARN de rol en la lista desplegable Rol de ejecución.
   d. Pegue el ARN en el ARN del rol.

- Utilice un rol con acceso de administrador: si tiene permisos para crear un rol con acceso total a todos los servicios y recursos AWS, puede usar ese rol para probar cualquier tipo de estado de su flujo de trabajo. Para ello, puede crear un rol de Step Functions servicio y añadirle la AdministratorAccess política en la IAM consola https://console.aws.amazon.com/iam/.

Control de errores

De forma predeterminada, cuando un estado registra un error, Step Functions hace que la ejecución del flujo de trabajo falle por completo. Para las acciones y algunos estados de flujo, puede configurar la forma en que Step Functions gestiona los errores. Incluso si ha configurado la gestión de errores, es posible que algunos errores provoquen un error en la ejecución del flujo de trabajo. Para obtener más información, consulte Control de errores en Step Functions. En Workflow Studio, configure el control de errores en la pestaña Control de errores del panel Inspector.



Reintentar en caso de errores

Puede añadir una o varias reglas a los estados de acción y al estado del flujo Parallel para volver a intentar la tarea cuando se produzca un error. Estas reglas se denominan reintentadores. Para añadir un reintentador, seleccione el icono de edición en el cuadro Reintentador #1 y, a continuación, configure sus opciones:

- (Opcional) Añada su comentario en el campo Comentario. No afectará al flujo de trabajo, pero se puede usar para anotarlo.
- Coloque el cursor en el campo Errores y elija un error que active el reintentador o introduzca un nombre de error personalizado. Puede elegir o añadir varios errores.
- (Opcional) Defina un intervalo. Se trata del tiempo en segundos que debe transcurrir antes de que Step Functions realice su primer reintento. Se realizarán reintentos adicionales a intervalos que puede configurar con el Máximo de intentos y la Tasa de regresión.
- (Opcional) Defina el Máximo de intentos. Este es el número máximo de reintentos antes de que Step Functions provoque un error en la ejecución.

- (Opcional) Establezca la Tasa de regresión. Se trata de un multiplicador que determina en qué medida aumentará el intervalo de reintentos con cada intento.

> **Note**
>
> No todas las opciones de control de errores están disponibles en todos los estados. Lambda Invoke tiene un reintentador configurado de forma predeterminada.

Detectar errores

Para detectar un error, se puede añadir una o varias reglas a los estados de acción y a los estados del flujo Parallel y Map. Estas reglas se denominan captadores. Para añadir un captador, seleccione Añadir nuevo captador y, a continuación, configure sus opciones:

- (Opcional) Añada su comentario en el campo Comentario. No afectará al flujo de trabajo, pero se puede usar para anotarlo.
- Coloque el cursor en el campo Errores y elija un error que active el captador o introduzca un nombre de error personalizado. Puede elegir o añadir varios errores.
- En el campo Estado alternativo, seleccione un estado alternativo. Este es el estado al que pasará el flujo de trabajo después de detectar un error.
- (Opcional) En el ResultPathcampo, añada un `ResultPath` filtro para añadir el error a la entrada de estado original. `ResultPath`Debe ser válido JsonPath. Este se enviará al estado alternativo.

Tiempos de espera

Puede configurar un tiempo de espera para los estados de acción para establecer el número máximo de segundos que el estado puede ejecutarse antes de que se produzca un error. Use tiempos de espera para evitar las ejecuciones bloqueadas. Para configurar un tiempo de espera, introduzca el número de segundos que su estado debe esperar antes de que se produzca un error en la ejecución. Para obtener más información acerca de los tiempos de espera, consulte `TimeoutSeconds` en estado Estado de la tarea.

HeartbeatSeconds

Puede configurar una notificación de latido o periódica enviada por su tarea. Si establece un intervalo de latidos y su estado no envía notificaciones de latidos en los intervalos configurados, la tarea se

marca como errónea. Para configurar un latido, establezca un número de segundos entero y positivo distinto a cero. Para obtener más información, consulte `HeartBeatSeconds` en estado Estado de la tarea.

Tutorial: Aprender a usar AWS Step Functions Workflow Studio

En este tutorial, aprenderá los conceptos básicos para trabajar con Workflow Studio para AWS Step Functions. En Modo Diseño de Workflow Studio, creará una máquina de estado que contenga varios estados `Pass`, incluidos `Choice`, `Fail`, `Wait` y `Parallel`. Utilizará la característica de arrastrar y soltar para buscar, seleccionar y configurar estos estados. A continuación, verá la definición generada automáticamente Lenguaje de estados de Amazon (ASL) de su flujo de trabajo. También puede editar la Modo Código de Workflow Studio para editar la definición del flujo de trabajo. A continuación, saldrá de Workflow Studio, ejecutará la máquina de estado y revisará los detalles de la ejecución.

En este tutorial, también aprenderá a actualizar la máquina de estado y a ver los cambios en el resultado de la ejecución. Por último, realizará un paso de limpieza y eliminará su máquina de estado.

Tras completar este tutorial, sabrá cómo usar Workflow Studio para crear y configurar un flujo de trabajo mediante los modos Diseño y Código. También sabrá cómo actualizar, ejecutar y eliminar su máquina de estado.

> **Note**
>
> Antes de empezar, asegúrate de cumplir los requisitos previos de este tutorial.

Temas

Paso 1: Navegar a Workflow Studio

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.

Paso 2: Crear una máquina de estado

En Workflow Studio, una máquina de estado es una representación gráfica del flujo de trabajo. Con Workflow Studio, puede definir, configurar y examinar los pasos individuales de su flujo de trabajo. En los siguientes pasos, utilice el Modo Diseño de Workflow Studio para crear su máquina de estado.

Para crear una máquina de estado

1. Asegúrese de estar en el modo de Diseño de Workflow Studio.
2. Desde el Navegador de estados de la parte izquierda, seleccione la pestaña Flujo. A continuación, arrastre el estado Pass al estado vacío con la etiqueta Arrastrar primero el estado aquí.
3. En la pestaña Flujo, arrastre y suelte un estado Choice debajo del estado Pass.
4. En Nombre del estado, sustituya el nombre predeterminado de Choice. Para este tutorial, use el nombre **`IsHelloWorldExample`**.
5. Arrastra otro estado de Pass y suéltalo en una sucursal del IsHelloWorldExampleestado. A continuación, arrastre un estado fallido y suéltelo debajo de la otra rama del IsHelloWorldExampleestado.
6. Elija el estado Pass (1) y cámbiele el nombre a **`Yes`**. Cambie el nombre del estado Fail a **`No`**.
7. Especifique la lógica de ramificación del IsHelloWorldExampleestado mediante la variable booleana. `IsHelloWorldExample`

   Si `IsHelloWorldExample` es `False`, el flujo de trabajo pasará al No. De lo contrario, el flujo de trabajo continuará su flujo de ejecución en el estado Sí.

   Para definir la lógica de ramificación, haga lo siguiente:

   a. Elija el IsHelloWorldExampleestado de la regla #1 yCanvas, a continuación, en Reglas de elección, elija el icono de edición del cuadro Regla para definir la primera regla de elección.

   b. Elija Añadir condiciones.

c. En el cuadro de diálogo Condiciones de la regla n.º 1, introduzca **`$.IsHelloWorldExample`**en Variable.

d. Elija es igual a en Operador.

e. Elija Constante booleana en Valor y, a continuación, elija verdadero en la lista desplegable.

f. Seleccione Guardar condiciones.

g. Asegúrese de que la lista desplegable El siguiente estado es: tenga seleccionada la opción Sí.

h. Seleccione Añadir nueva regla de elección y, a continuación, seleccione Añadir condiciones.

i. En el cuadro Regla n.º 2, defina la segunda regla de elección cuando el valor de la variable `IsHelloWorldExample` sea falso repitiendo los subpasos 7.c a 7.f. Para el paso 7.e, elija falso en lugar de verdadero.

j. En el cuadro Regla n.º 2, seleccione No en la lista desplegable El siguiente estado es:.

k. En el cuadro Regla predeterminada, elija el icono de edición para definir la regla de elección predeterminada y, a continuación, elija Sí en la lista desplegable.

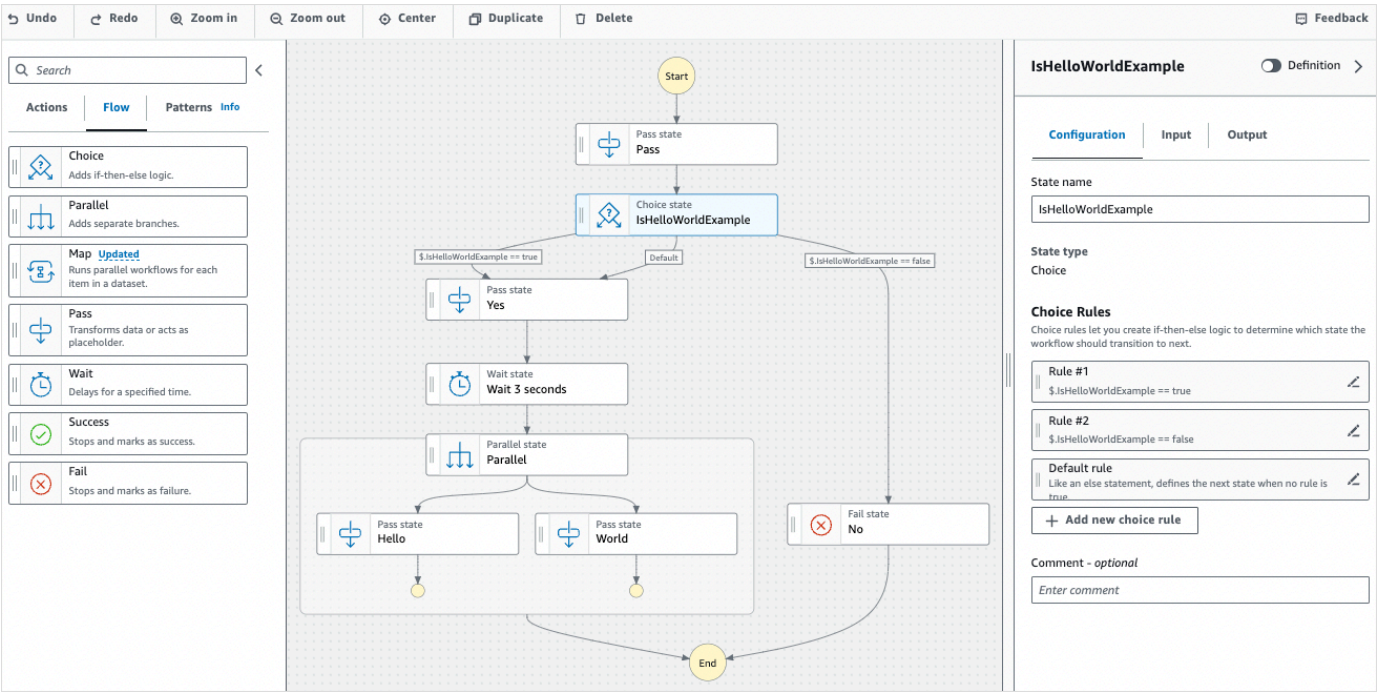
8. Añada un estado Wait después del estado Sí y asígnele el nombre **`Wait 3 sec`**. A continuación, configure el tiempo de espera para que sea de tres segundos siguiendo estos pasos:

   a. En Opciones, mantenga la selección predeterminada de Esperar un intervalo fijo.

   b. En Segundos, asegúrese de que esté seleccionada la opción Introducir segundos y, a continuación, introduzca **`3`** en el cuadro.

9. Tras el estado Wait 3 sec, añada un estado Parallel. Añada dos estados Pass en sus dos ramificaciones. Indique el primer estado de Pass **`Hello`**. Indique el segundo estado Pass **`World`**.

   El flujo de trabajo completo será como el siguiente:



Paso 3: Revisar la definición de Amazon States Language generada automáticamente

A medida que arrastra y suelta los estados de la pestaña Flujo al lienzo, Workflow Studio elabora automáticamente la definición de su flujo de trabajo en Amazon States Language (ASL) en tiempo real. En el panel de Inspector, pulse el botón de alternancia Definición para ver esta definición o cambie al Modo Código para editar esta definición según sea necesario. Para obtener información sobre la edición de la definición del flujo de trabajo, consulte el Paso 4 de este tutorial.

- (Opcional) Seleccione Definición en el panel Inspector y visualice el flujo de trabajo de la máquina de estado.

  El siguiente código de ejemplo muestra la definición autogenerada de Amazon States Language para la máquina de estado `IsHelloWorldExample`. El estado `Choice` que agregó en Workflow Studio se usa para determinar el flujo de ejecución en función de la lógica de ramificación definida en el Paso 2.

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

Paso 4: Editar la definición del flujo de trabajo en el modo Código

El modo Código de Workflow Studio ofrece un editor de código integrado para ver, y editar la definición de ASL de sus flujos de trabajo.

1. Elija Código para cambiar al modo Código.
2. Tras la definición del estado Parallel, coloque el cursor y pulse **`Enter`**.
3. Pulse **`Ctrl+space`** para ver la lista de estados que puede añadir después del estado Parallel.
4. Elija Estado Pass en la lista de opciones. El editor de código añade un código reutilizable para el Estado Pass.
5. La adición de este estado provoca errores en la definición del flujo de trabajo. En la definición del estado Parallel, sustituya `"End": true` por **`"Next": "PassState"`**.
6. En la definición de Estado Pass que agregó, realice los siguientes cambios:
   a. Elimine el nodo Resultado.
   b. Elimine `"ResultPath": "$.result",` y `"Next": "NextState"`.
   c. Después de `"Type": "Pass",`, introduzca **`"End": true`**.

d. Añada una **,** después de la definición de Estado Pass.

La definición de del flujo de trabajo debería ser similar a la definición siguiente.

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

Paso 5: Guardar la máquina de estado

1. Elija el modo Config o elija el icono de edición junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de la máquina de estado, especifique un nombre. Por ejemplo, introduzca **`HelloWorld`**.
2. (Opcional) Especifique otra configuración del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución. Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.
3. Seleccione Crear.
4. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede elegir Ver configuración de roles para volver al modo Config.

Para obtener más información sobre el modo Config, consulte Modo Config de Workflow Studio.

Paso 6: Ejecutar la máquina de estado

Las ejecuciones de máquinas de estado son instancias en las que se ejecuta un flujo de trabajo para realizar tareas.

1. En la página Máquinas de estado, elija la máquina de HelloWorldestado.
2. En la HelloWorldpágina, elija Iniciar ejecución.
3. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

4. En el cuadro Entrada, introduzca los valores de entrada para la ejecución en formato JSON. En función de su entrada, la variable `IsHelloWorldExample` determina en qué estado se ejecutará el flujo de la máquina. Por ahora, introduzca el siguiente valor de entrada:

   ```
   {
      "IsHelloWorldExample": true
   }
   ```

   > **Note**
   >
   > Aunque especificar una entrada de ejecución es opcional, en este tutorial es obligatorio especificar una entrada de ejecución similar a la entrada del ejemplo anterior. Se hace referencia a este valor de entrada en el estado `Choice` al ejecutar la máquina de estado.

5. Seleccione Iniciar ejecución.

6. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

   En este tutorial, si ha introducido un valor de entrada de `"IsHelloWorldExample": true`, debería ver el siguiente resultado:

```
{
   "IsHelloWorldExample": true
},
{
   "IsHelloWorldExample": true
}
```

Paso 7: Actualizar la máquina de estado

Cuando actualices una máquina de estado, sus actualizaciones son finalmente consistentes. Tras un breve periodo de tiempo, todas las ejecuciones recién iniciadas reflejarán la definición actualizada de la máquina de estado. Según la definición anterior, todas las ejecuciones que se estén ejecutando actualmente se ejecutarán hasta completarse.

En este paso, actualizará su máquina de estado en el modo Modo Diseño de Workflow Studio. Añadirá un campo `Result` en el estado Pass denominado World.

1. En la página titulada con su ID de ejecución, seleccione Editar máquina de estado.
2. Asegúrese de estar en el modo Diseño.
3. Elija el estado Pass llamado World en el lienzo y, a continuación, seleccione Salida.
4. En el cuadro Resultado, introduzca **`"World has been updated!"`**.
5. Seleccione Guardar.
6. (Opcional) En el área Definición, consulte la definición actualizada de su flujo de trabajo en Amazon States Language.

```
{
      "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. Elija Ejecutar.
8. En el cuadro de diálogo Iniciar la ejecución, que se abre en una nueva pestaña, introduzca la siguiente entrada de ejecución.

```
{
   "IsHelloWorldExample": true
}
```

9. Seleccione Iniciar ejecución.
10. (Opcional) En la Vista gráfica, elija el paso World y, a continuación, elija Salida. El resultado es ¡World se ha actualizado!

Paso 8: Eliminación

Para eliminar su máquina de estado

1. En el panel de navegación, elija Máquinas de estados.
2. En la página Máquinas de estado, seleccione y HelloWorld, a continuación, elija Eliminar.
3. En el cuadro de diálogo Eliminar máquina de estado, escriba **`delete`** para confirmar la eliminación.
4. Elija Eliminar.

   Si la eliminación se ha realizado correctamente, aparecerá una barra verde en la parte superior de la pantalla. La barra de estado verde indica que la máquina de estado está marcada para su eliminación. La máquina de estado se eliminará cuando todas las ejecuciones en curso dejen de ejecutarse.

Para eliminar el rol de ejecución

1. Abra la página de roles de IAM.
2. Elija el rol de IAM, que Step Functions creó para usted. Por ejemplo, StepFunctions- HelloWorld -Role-Example.
3. Elija Eliminar rol.
4. Elija Sí, eliminar.

Tutoriales de Step Functions

Los tutoriales de esta sección pueden ayudarle a comprender los distintos aspectos de trabajar con AWS Step Functions.

Para completar estos tutoriales, necesita una AWS cuenta. Si no tiene una AWS cuenta, vaya a https://aws.amazon.com/ y elija Crear una AWS cuenta.

Temas

Creación de una máquina de estado de Step Functions que utilice Lambda

En este tutorial, creará un flujo de trabajo de un solo paso AWS Step Functions para invocar una AWS Lambda función.

> **Note**
>
> Step Functions se basa en máquinas y tareas de estados. En Step Functions, las máquinas de estados se denominan flujos de trabajo, que son una serie de pasos basados en eventos. Cada paso de un flujo de trabajo se denomina estado. Por ejemplo, el estado de una tarea representa una unidad de trabajo que realiza otro AWS servicio, como llamar a otro Servicio de AWS o a una API.
> Para obtener más información, consulte:
>
> - ¿Qué es AWS Step Functions?
> - Llama a otros AWS servicios

Lambda es adecuado para estados de `Task`, ya que las funciones de Lambda son sin servidor y fáciles de escribir. Puedes escribir código en el editor AWS Management Console o en tu editor favorito. AWS se ocupa de los detalles necesarios para proporcionar un entorno informático para su función y para ejecutarla.

En este tema:

Paso 1: Crear una función de Lambda

La función de Lambda recibe los datos de evento y devuelve un mensaje de bienvenida.

> **Important**
>
> Asegúrese de que su función Lambda esté en la misma AWS cuenta y AWS región que su máquina de estado.

1. Abra la consola de Lambda; y elija Crear función.
2. En la página Crear función, elija Diseñar desde cero.

3. En Nombre de la función, introduzca `HelloFunction`.
4. Mantenga las selecciones predeterminadas para todas las demás opciones y elija Crear función.
5. Tras crear la función de Lambda, copie el nombre de recurso de Amazon (ARN) de la función que aparece en la esquina superior derecha de la página. Para copiar el ARN, haga clic en  .

   A continuación se muestra un ejemplo de ARN:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
   ```
6. Copie el siguiente código para la función Lambda en la sección Código fuente de la ***HelloFunction***página.

   ```
   export const handler = async(event, context, callback) => {
       callback(null, "Hello from " + event.who + "!");
   };
   ```

   Este código crea un saludo utilizando el campo `who` de los datos de entrada, proporcionados por el objeto `event` pasado a la función. Los datos de entrada de esta función se agregan más tarde, cuando se inicia una nueva ejecución. El método `callback` devuelve el saludo creado a partir de su función.
7. Elija Implementar.

Paso 2: Probar la función de Lambda

Pruebe la función de Lambda para verla en acción.

1. Seleccione Probar.
2. En Nombre del evento, escriba `HelloEvent`.
3. Sustituya los datos de Evento JSON por lo siguiente.

   ```
   {
       "who": "AWS Step Functions"
   }
   ```

   La entrada `"who"` se corresponde con el campo `event.who` de su función de Lambda y completa el saludo. Introducirá los mismos datos de entrada cuando ejecute su máquina de estado.

4. Elija Guardar y, a continuación, Probar.
5. En Resultado de ejecución, expanda Detalles para ver los resultados.

Paso 3: Crear una máquina de estado

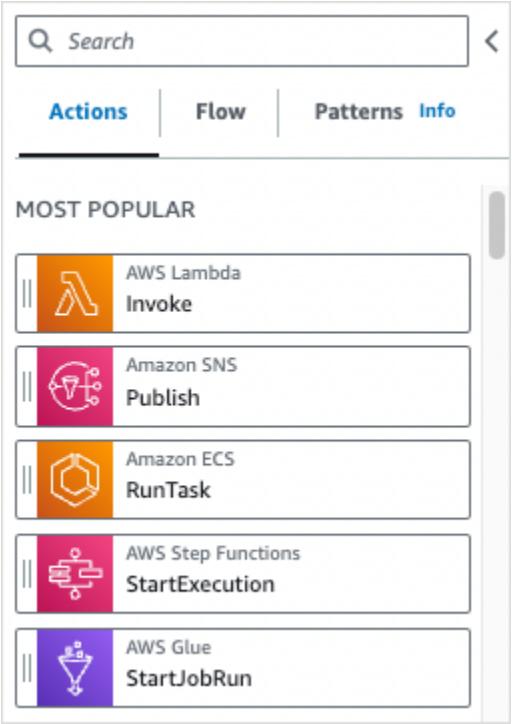
Utilice la consola de Step Functions para crear una máquina de estado que invoque la función de Lambda que creó en el paso 1.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

   > **Important**
   >
   > Asegúrese de que su máquina de estado esté en la misma AWS cuenta y región que la función Lambda que creó anteriormente.

2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. En el navegador de estados de la izquierda, asegúrese de que ha elegido la pestaña Acciones. A continuación, proceda del modo siguiente:

   - Arrastre y suelte la API Invocación de AWS Lambda en el estado vacío con la etiqueta Arrastrar primer estado aquí.



5. En el panel Inspector de la derecha, configure la función de Lambda:

   a. En la sección Parámetros de la API, elija la función de Lambda que creó anteriormente en la lista desplegable Nombre de la función.

   b. Mantenga la selección predeterminada en la lista desplegable de Carga útil.

6. (Opcional) Seleccione Definición para ver la definición de Lenguaje de estados de Amazon (ASL) de la máquina de estado, que se genera automáticamente en función de las selecciones que realice en la pestaña Acciones y el panel Inspector.

7. Especifique un nombre para la máquina de estado. Para ello, elija el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   Por ejemplo, introduzca el nombre **`LambdaStateMachine`**.

   > **Note**
   >
   > Los nombres de máquinas de estado, ejecuciones y tareas de actividad no deben superar los 80 caracteres. Estos nombres deben ser exclusivos para su cuenta y AWS región, y no deben contener ninguno de los siguientes elementos:

- Espacios en blanco
- Caracteres comodín (? *)
- Caracteres entre corchetes (< > { } [ ])
- Caracteres especiales (" # % \ ^ | ~ ` $ & , ; : /)
- Caracteres de control (\\u0000 - \\u001f o \\u007f - \\u009f).

Si la máquina de estado es de tipo rápido, puede proporcionar el mismo nombre a varias ejecuciones de la máquina de estado. Step Functions genera un ARN de ejecución único para cada ejecución de una máquina de estado rápida, incluso aunque varias ejecuciones tengan el mismo nombre.
Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

8. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

9. Seleccione Crear.
10. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

    También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 4: Ejecutar la máquina de estado

Después de crear la máquina de estado, puede ejecutarla.

1. En la página Máquinas de estado, elija LambdaStateMachine.
2. Seleccione Iniciar ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.
3. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.
4. En el área Entrada, sustituya los datos de ejecución de ejemplo por lo siguiente.

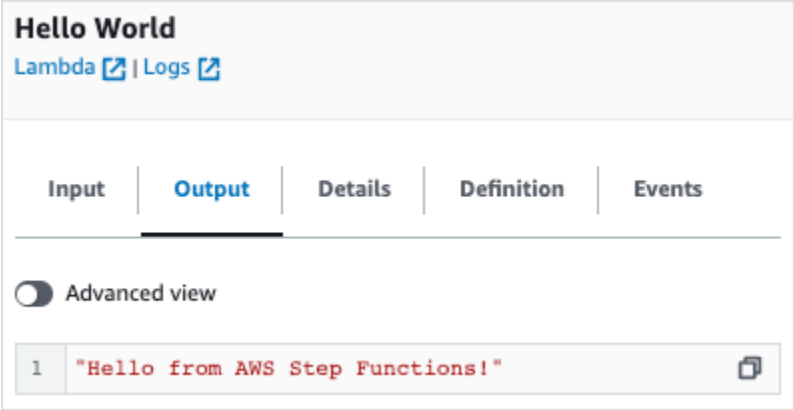
   ```
   {
       "who" : "AWS Step Functions"
   }
   ```

   `"who"` es el nombre de clave que su función de Lambda utiliza para obtener el nombre de la persona a la que se saluda.
5. Seleccione Iniciar ejecución.

   Se inicia la ejecución de su máquina de estado y aparece una nueva página que muestra la ejecución en funcionamiento.
6. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más

información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.



> **Note**
>
> También puede transferir cargas útiles al invocar a Lambda desde una máquina de estado. Para obtener más información y ejemplos sobre cómo invocar a Lambda pasando la carga útil en el campo `Parameters`, consulte Invocar Lambda con Step Functions.

Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos

En este tutorial, creará una máquina de AWS Step Functions estados con un Estados alternativos campo. El `Catch` campo usa una AWS Lambda función para responder con una lógica condicional basada en el tipo de mensaje de error. Esta técnica se denomina control de errores de funciones.

Para obtener más información, consulte Errores de la función AWS Lambda en Node.js en la Guía para desarrolladores de AWS Lambda .

> **Note**
>
> También puede crear máquinas de estado que reintenten en los tiempos de espera o que usen `Catch` para adoptar un estado específico cuando se produce un error o se agota un tiempo de espera. Para ver ejemplos de estas técnicas de control de errores, consulte Ejemplos sobre el uso de Retry y Catch.

En este tema:

Paso 1: Crear una función de Lambda que no funciona correctamente

Utilice una función de Lambda para simular una condición de error.

> **Important**
>
> Asegúrese de que su función Lambda esté en la misma AWS cuenta y AWS región que su máquina de estado.

1. Abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. Elija Crear función.
3. Seleccione Usar un esquema, introduzca `step-functions` en el cuadro de búsqueda y, a continuación, elija el esquema Lanzar un error personalizado.
4. En Nombre de la función, introduzca `FailFunction`.
5. En Rol, mantenga la selección predeterminada (Crear un nuevo rol con permisos básicos de Lambda).
6. El código siguiente aparece en el panel Código de función de Lambda.

```
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!');
};
```

El objeto `context` devuelve el mensaje de error `This is a custom error!`

7. Seleccione Crear función.
8. Tras crear la función de Lambda, copie el nombre de recurso de Amazon (ARN) de la función que aparece en la esquina superior derecha de la página. Para copiar el ARN, haga clic en  .

   A continuación se muestra un ejemplo de ARN:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:FailFunction
   ```
9. Elija Implementar.

Paso 2: Probar la función de Lambda

Pruebe la función de Lambda para verla en acción.

1. En la FailFunctionpágina, seleccione la pestaña Probar y, a continuación, elija Probar. No es necesario crear un evento de prueba.
2. Para revisar los resultados de la prueba (el error simulado), en Resultado de la ejecución, expanda Detalles.

Paso 3: Crear una máquina de estado con un campo Catch

Utilice la consola de Step Functions para crear una máquina de estado que utilice un estado Estado de la tarea con un campo `Catch`. Agregue una referencia a la función de Lambda en el estado Tarea. La máquina de estado invoca la función de Lambda, que falla durante la ejecución. Step Functions reintenta la función dos veces utilizando un retroceso exponencial entre los reintentos.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Elija la pestaña Código para abrir el editor de código. En el editor de código también puede escribir y editar la definición Lenguaje de estados de Amazon de ASL de los flujos de trabajo.
5. Pegue el siguiente código, pero sustituya el ARN de la función de Lambda que creó anteriormente en el campo `Resource`.

   ```
   {
   ```

```
  "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
     }
  }
}
```

Esta es una descripción de la máquina de estado realizada que utiliza el Amazon States Language. Define un único estado `Task` llamado `CreateAccount`. Para obtener más información, consulte Estructura de las máquinas de estado.

Para obtener más información acerca de la sintaxis del campo `Retry`, consulte Ejemplos de máquina de estado que usan Retry y Catch.

> **Note**
>
> Los errores no controlados en Lambda se notifican como `Lambda.Unknown` en el resultado del error. Entre ellos se incluyen out-of-memory los errores y los tiempos de espera de las funciones. Puede buscar coincidencias de estos errores con `Lambda.Unknown`, `States.ALL` o `States.TaskFailed` para controlarlos. Cuando Lambda alcanza el número máximo de invocaciones, el error es. `Lambda.TooManyRequestsException` Para obtener más información acerca de errores de función de Lambda, consulte Tratamiento de errores y reintentos automáticos en la Guía para desarrolladores de AWS Lambda .

6. (Opcional) En el Panel de visualización de gráficos, consulte la visualización gráfica en tiempo real de su flujo de trabajo.
7. Especifique un nombre para la máquina de estado. Para ello, seleccione el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, escriba **`Catchfailure`**.
8. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.
9. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 4: Ejecutar la máquina de estado

Después de crear la máquina de estado, puede ejecutarla.

1. En la página Máquinas de estados, elija Catchfailure.
2. En la página Catchfailure, seleccione Iniciar ejecución. Aparece el cuadro de diálogo Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

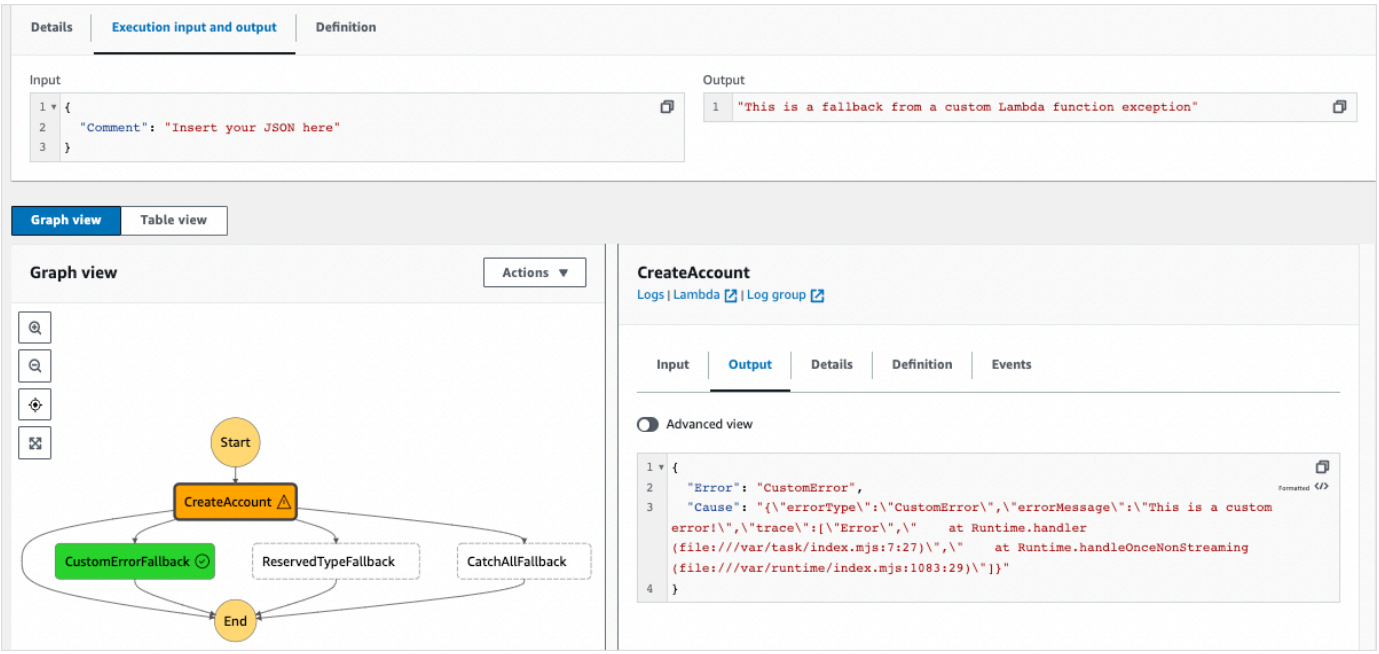
   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.
   3. Seleccione Iniciar ejecución.
   4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

   Por ejemplo, para ver su mensaje de error personalizado, elija el CreateAccountpaso en la vista de gráfico y, a continuación, elija la pestaña Salida.



> **Note**
>
> Puede conservar la entrada del estado con el error utilizando `ResultPath`. Consulte Se utiliza ResultPath para incluir tanto el error como la entrada en un `Catch`.

Uso del estado Inline Map para repetir una acción

Este tutorial puede ser de ayuda para familiarizarse con el estado `Map` en modo En línea. El estado Inline Map se utiliza en los flujos de trabajo para realizar una acción repetidamente. Para obtener más información sobre el modo en línea, consulte Estado Map en modo En línea.

En este tutorial, utilizará el estado Inline Map para generar repetidamente los identificadores únicos universales de la versión 4 (UUID v4). Comience por crear un flujo de trabajo que contenga dos estados Pass y un estado Inline Map en Workflow Studio. A continuación, configure la entrada y la salida, incluida la matriz JSON de entrada para el estado `Map`. El estado `Map` devuelve una matriz de salida que contiene los UUID v4 generados para cada elemento de la matriz de entrada.

Contenido

- Paso 3: Revisar la definición de Amazon States Language generada automáticamente y guardar el flujo de trabajo
- Paso 4: Ejecutar la máquina de estado

Paso 1: Crear el prototipo de flujo de trabajo

En este paso, creará el prototipo para el flujo de trabajo de utilizando Workflow Studio. Workflow Studio es un diseñador visual de flujos de trabajo disponible en la consola de Step Functions. Elegirá los estados necesarios en la pestaña Flujo y utilizará la característica de arrastrar y soltar de Workflow Studio para crear el prototipo del flujo de trabajo.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Desde la pestaña Flujo, arrastre un estado Pass y suéltelo en el estado vacío denominado Arrastrar primero el estado aquí.
5. Arrastre un estado Map y suéltelo debajo del estado Pass. Cambie el nombre del estado Map a **`Map demo`**.
6. Arrastre un segundo estado Pass y colóquelo dentro del estado Map demo.
7. Cambie el nombre del segundo estado Pass a **`Generate UUID`**.

Paso 2: Configurar entrada y salida

En este paso, configurará la entrada y la salida para todos los estados de su prototipo de flujo de trabajo. En primer lugar, se inyectan algunos datos fijos en el flujo de trabajo mediante el primer estado Pass. Este estado Pass transfiere estos datos como entrada al estado Map demo. En esta entrada, se especifica el nodo que contiene la matriz de entrada sobre la que debe iterarse el estado Map demo. A continuación, defina el paso que debe repetir el estado Map demo para generar los UUID de la versión 4. Por último, configure la salida para que regrese para cada repetición.

1. Elija el primer estado Pass en su prototipo de flujo de trabajo. En la pestaña Salida, introduzca lo siguiente en Resultado:

```
{
  "foo": "bar",
  "colors": [
```

```
    "red",
    "green",
    "blue",
    "yellow",
    "white"
  ]
}
```

2. Seleccione el estado Map demo y, en la pestaña Configuración, haga lo siguiente:

   a. Seleccione Proporcionar una ruta a la matriz de elementos.

   b. Especifique la siguiente ruta de referencia para seleccionar el nodo que contiene la matriz de entrada:

   ```
   $.colors
   ```

3. Seleccione el estado Generate UUID y, en la pestaña Entrada, haga lo siguiente:

   a. Elija Transformar la entrada con parámetros.

   b. Introduzca la siguiente entrada JSON para generar los UUID de la versión 4 para cada uno de los elementos de la matriz de entrada. Se utiliza la función intrínseca `States.UUID` para generar los UUID.

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

4. Para el estado Generate UUID, seleccione la pestaña Salida y haga lo siguiente:

   a. Elija Filtrar salida con OutputPath.

   b. Introduzca la siguiente ruta de referencia para seleccionar el nodo JSON que contiene los elementos de la matriz de salida:

   ```
   $.uuid
   ```

Paso 3: Revisar la definición de Amazon States Language generada automáticamente y guardar el flujo de trabajo

A medida que arrastra y suelta los estados del panel Flujo al lienzo, Workflow Studio elabora automáticamente la definición de su flujo de trabajo en Amazon States Language (ASL) en tiempo real. Puede editar esta definición según sea necesario.

1. (Opcional) Seleccione Definición en el panel de Inspector para ver la definición del flujo de trabajo de Amazon States Language generada automáticamente.

   > **Tip**
   >
   > También puede ver la definición de ASL en el Editor de código de Workflow Studio. En el editor de código también puede editar la definición de ASL del flujo de trabajo.

   El siguiente ejemplo muestra la definición de Amazon States Language generada automáticamente para su flujo de trabajo.

```
{
    "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. Especifique un nombre para la máquina de estado. Para ello, seleccione el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`InlineMapDemo`**.

3. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

4. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de

las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 4: Ejecutar la máquina de estado

Las ejecuciones de máquinas de estado son instancias en las que se ejecuta un flujo de trabajo para realizar tareas.

1. En la InlineMapDemopágina, elija Iniciar ejecución.
2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.
   3. Seleccione Iniciar ejecución.
   4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Para ver la entrada y la salida de la ejecución side-by-side, seleccione Entrada y salida de ejecución. En Salida, consulte la matriz de salida devuelta por el estado `Map`. A continuación se muestra un ejemplo de la matriz de salida:

```
[
  "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"
]
```

Copiar datos CSV a gran escala mediante Distributed Map

Este tutorial le ayuda a empezar a utilizar el estado `Map` en modo distribuido. Un estado `Map` establecido en Distributed se conoce como estado Map Distributed. El estado Distributed Map se utiliza en los flujos de trabajo para iterar sobre orígenes de datos de Amazon S3 a gran escala. El estado `Map` ejecuta cada iteración como una ejecución de flujo de trabajo secundario, lo que permite una alta simultaneidad. Para obtener más información sobre el modo distribuido, consulte Estado Map en modo distribuido.

En este tutorial, utilice el estado Distributed Map para recorrer en iteración un archivo CSV en un bucket de Amazon S3. A continuación, devuelve su contenido, junto con el ARN de la ejecución de un flujo de trabajo secundario, en otro bucket de Amazon S3. Comience por crear un prototipo de flujo de trabajo en Workflow Studio. A continuación, establezca el modo de procesamiento del estado `Map` en Distribuido, especifique el archivo CSV como conjunto de datos y proporcione su ubicación al estado `Map`. También debe especificar el tipo de flujo de trabajo para las ejecuciones del flujo de trabajo secundario en las que el estado Distributed Map comienza como Rápido.

Además de estos ajustes, también debe especificar otras configuraciones, como el número máximo de ejecuciones simultáneas de flujos de trabajo secundarios y la ubicación para exportar el resultado de `Map`, para el flujo de trabajo de ejemplo utilizado en este tutorial.

Contenido

- Paso 2: Configurar los campos necesarios para el estado Map
- Paso 3: Configurar opciones adicionales
- Paso 4: Configurar la función de Lambda
- Paso 5: Actualizar el prototipo de flujo de trabajo
- Paso 6: Revisar la definición de Amazon States Language generada automáticamente y guardar el flujo de trabajo
- Paso 7: Ejecutar la máquina de estado

Requisitos previos

- Cargar un archivo CSV en un bucket de Amazon S3. Debe definir una fila de encabezado en el archivo CSV. Para obtener información sobre los límites de tamaño impuestos al archivo CSV y cómo especificar la fila de encabezado, consulte Archivo CSV en un bucket de Amazon S3.
- Cree otro bucket de Amazon S3 y una carpeta dentro del mismo para exportar el resultado del estado `Map`.

> **Important**
>
> Asegúrese de que los buckets de Amazon S3 estén en la misma máquina de estado Cuenta de AWS y en la misma posición Región de AWS que ella.

Paso 1: Crear el prototipo de flujo de trabajo

En este paso, creará el prototipo para el flujo de trabajo de utilizando Workflow Studio. Workflow Studio es un diseñador visual de flujos de trabajo disponible en la consola de Step Functions. Puede elegir el estado y la acción de API necesarios en las pestañas Flujo y Acciones, respectivamente. Utilizará la característica de arrastrar y soltar de Workflow Studio para crear el prototipo del flujo de trabajo.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Desde la pestaña Flujo, arrastre un estado Map al estado vacío con la etiqueta Arrastrar la primera acción aquí.

5. En la pestaña Configuración, en Nombre de estado, escriba **`Process data`**.
6. En la pestaña Acciones, arrastre una acción de la API Invocación de AWS Lambda y suéltela dentro del estado Procesar datos.
7. Cambie el nombre del estado Invoke AWS Lambda a **`Process CSV data`**.

Paso 2: Configurar los campos necesarios para el estado Map

En este paso, configurará los siguientes campos obligatorios del estado Distributed Map:

- ItemReader— Especifica el conjunto de datos y su ubicación desde la que el `Map` estado puede leer la entrada.
- ItemProcessor: especifica los siguientes valores:
  - `ProcessorConfig`, defina los valores `Mode` y `ExecutionType` en `DISTRIBUTED` y`EXPRESS`, respectivamente. Esto establece el modo de procesamiento del estado `Map` y el tipo de flujo de trabajo para las ejecuciones de flujos de trabajo secundarios en las que se inicia el estado Distributed Map.
  - `StartAt`: el primer estado del flujo de trabajo de Map.
  - `States`: define el flujo de trabajo de Map, que consiste en un conjunto de pasos que se deben repetir en cada ejecución del flujo de trabajo secundario.
- ResultWriter— Especifica la ubicación de Amazon S3 en la que Step Functions escribe los resultados del estado del mapa distribuido.

> **Important**
>
> Asegúrese de que el depósito de Amazon S3 que utiliza para exportar los resultados de una ejecución de mapas esté en la misma máquina de estados Cuenta de AWS y Región de AWS que esté en su máquina de estados. De lo contrario, la ejecución de la máquina de estado fallará y se producirá el error `States.ResultWriterFailed`.

Para configurar los campos necesarios:

1. Elija el estado Procesar datos y haga lo siguiente en la pestaña Configuración; haga lo siguiente:
   a. Para Modo de procesamiento, elija Distribuido.

b. En Origen del elemento, elija Amazon S3 y, a continuación, elija el Archivo CSV en S3) de la lista desplegable Origen del elemento de S3.

c. Haga lo siguiente para especificar la ubicación en Amazon S3 de su archivo CSV:

   i. Para Objeto S3, seleccione Introducir bucket y clave en la lista desplegable.

   ii. En Bucket, introduzca el nombre del bucket de Amazon S3, que contiene el archivo CSV. Por ejemplo, **`sourceBucket`**.

   iii. En Clave, introduzca el nombre del objeto de Amazon S3 en el que guardó el archivo CSV. También debe especificar el nombre del archivo CSV en este campo. Por ejemplo, **`csvDataset/ratings.csv`**.

d. Para los archivos CSV, también debe especificar la ubicación del encabezado de la columna. Para ello, elija Configuración adicional y, a continuación, para la Ubicación del encabezado CSV, mantenga la selección predeterminada de Primera fila si la primera fila del archivo CSV es el encabezado. De lo contrario, elija Dado para especificar el encabezado dentro de la definición de la máquina de estado. Para obtener más información, consulte `ReaderConfig`.

e. Para Tipo de ejecución secundaria, elija Express.

2. En Exportar ubicación, para exportar los resultados de Map Run a una ubicación específica de Amazon S3, elija Exportar la salida del estado Map a Amazon S3.

3. Haga lo siguiente:

   a. Para Bucket de S3, seleccione Introducir el nombre y el prefijo del bucket en la lista desplegable.

   b. En bucket, escriba el nombre del bucket de Amazon S3 en el que desea exportar los resultados. Por ejemplo, **`mapOutputs`**.

   c. En Prefijo, introduzca el nombre de la carpeta en la que desee guardar los resultados. Por ejemplo, **`resultData`**.

Paso 3: Configurar opciones adicionales

Además de los ajustes necesarios para un estado Distributed Map, también puede especificar otras opciones. Estas pueden incluir el número máximo de ejecuciones simultáneas de flujos de trabajo secundarios y la ubicación a la que se debe exportar el resultado del estado `Map`.

1. Seleccione el estado Procesar datos. A continuación, en Fuente de elemento, elija Configuración adicional.
2. Haga lo siguiente:

   a. Seleccione Modificar elementos con ItemSelector para especificar una entrada JSON personalizada para cada ejecución de flujo de trabajo secundario.

   b. Introduzca la entrada de JSON siguiente:

   ```
   {
     "index.$": "$$.Map.Item.Index",
     "value.$": "$$.Map.Item.Value"
   }
   ```

   Para obtener información sobre cómo crear una entrada personalizada, consulte `ItemSelector`.
3. En la Configuración de versión ejecutable, en Límite de simultaneidad, especifique el número de ejecuciones simultáneas del flujo de trabajo secundario que puede iniciar el estado Distributed Map. Por ejemplo, escriba **`100`**.
4. Abra una nueva ventana o pestaña en el navegador y complete la configuración de la función de Lambda que utilizará en este flujo de trabajo, tal y como se explica en Paso 4: Configurar la función de Lambda.

Paso 4: Configurar la función de Lambda

> **Important**
>
> Asegúrese de que su función Lambda esté en la Región de AWS misma posición que su máquina de estados.

1. Abra la consola de Lambda; y elija Crear función.
2. En la página Crear función, elija Diseñar desde cero.
3. En la sección Información básica, configure la función de Lambda:

   a. En Nombre de la función, introduzca **`distributedMapLambda`**.

   b. En Tiempo de ejecución, elija Node.js 16.x.

c. Mantenga todas las selecciones predeterminadas y elija Crear función.

d. Tras crear la función de Lambda, copie el Nombre de recurso de Amazon (ARN) de la función que aparece en la esquina superior derecha de la página. Deberá proporcionar esto en su prototipo de flujo de trabajo. Para copiar el ARN, haga clic en  .

A continuación se muestra un ejemplo de ARN:

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

4. Copie el siguiente código para la función Lambda y péguelo en la sección Código fuente de la distributedMapLambdapágina.

```
exports.handler = async function(event, context) {
    console.log("Received Input:\n", event);

    return {
        'statusCode' : 200,
        'inputReceived' : event //returns the input that it received
    }
};
```

5. Elija Implementar. Una vez implementada la función, elija Probar para ver el resultado de la función de Lambda.

Paso 5: Actualizar el prototipo de flujo de trabajo

En la consola de Step Functions, actualizará su flujo de trabajo para añadir el ARN de la función de Lambda.

1. Vuelva a la pestaña o ventana en la que creó el prototipo del flujo de trabajo.
2. Elija el estado Procesar datos de CSV y haga lo siguiente en la pestaña Configuración; haga lo siguiente:

   a. En Tipo de integración, elija Optimizado.

   b. En Función de Lambda, empiece a introducir el nombre de la función de Lambda. Elija la función en la lista desplegable que aparece o elija Introducir nombre de función e introduzca el ARN de la función de Lambda.

Paso 6: Revisar la definición de Amazon States Language generada automáticamente y guardar el flujo de trabajo

A medida que arrastra y suelta los estados de las pestañas Acción y Flujo al lienzo, Workflow Studio redacta automáticamente la definición de su flujo de trabajo en Amazon States Language en tiempo real. Puede editar esta definición según sea necesario.

1. (Opcional) Seleccione Definición en el Inspector panel y visualice la definición de la máquina de estado.

   > **Tip**
   >
   > También puede ver la definición de ASL en el Editor de código de Workflow Studio. En el editor de código también puede editar la definición de ASL del flujo de trabajo.

   El siguiente código de ejemplo muestra la definición de Amazon States Language generada automáticamente para su flujo de trabajo.

```
{
  "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. Especifique un nombre para la máquina de estado. Para ello, seleccione el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`DistributedMapDemo`**.

3. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

4. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

   > **Note**
   >
   > Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 7: Ejecutar la máquina de estado

Una ejecución es una instancia de su máquina de estado en la que ejecuta su flujo de trabajo para realizar tareas.

1. En la DistributedMapDemopágina, elija Iniciar ejecución.
2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Por ejemplo, elija el estado `Map` y, a continuación, elija Map Run para abrir la página de Detalles de Map Run. En esta página, puede ver todos los detalles de ejecución de estado Map Distributed y las ejecuciones del flujo de trabajo secundario que inició. Para obtener información acerca de esta página, consulte Examen de Map Run.

Procesamiento de lotes completos de datos con una función de Lambda

En este tutorial, utilizará el campo ItemBatcher de estado Distributed Map para procesar un lote completo de elementos dentro de una función de Lambda. Cada lote contiene un máximo de tres elementos. El estado Distributed Map inicia cuatro ejecuciones de flujos de trabajo secundarios, en las que cada ejecución procesa tres elementos, mientras que una ejecución procesa un solo elemento. Cada ejecución del flujo de trabajo secundario invoca una función de Lambda que itera sobre los elementos individuales presentes en el lote.

Creará una máquina de estado que realiza la multiplicación de una matriz de números enteros. Supongamos que la matriz de enteros que proporciona como entrada es `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` y el factor de multiplicación es 7. Entonces, la matriz resultante formada después de multiplicar estos enteros por un factor de 7, será `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`.

Temas

- Paso 1: Crear la máquina de estado
- Paso 2: Crear la función de Lambda

- Paso 3: Ejecutar la máquina de estado

Paso 1: Crear la máquina de estado

En este paso, creará el prototipo de flujo de trabajo de la máquina de estado que pasa un lote completo de datos a la función de Lambda que creará en el Paso 2.

- Utilice la siguiente definición para crear una máquina de estado mediante la consola de Step Functions. Para obtener información sobre cómo crear una máquina de estado, consulte Paso 1: Crear el prototipo de flujo de trabajo en el tutorial Cómo empezar a usar el estado Distributed Map.

  En esta máquina de estado, se define un estado Distributed Map que acepta una matriz de 10 enteros como entrada y pasa esta matriz a una función de Lambda en lotes de 3. La función de Lambda itera sobre los elementos individuales presentes en el lote y devuelve una matriz de salida denominada `multiplied`. La matriz de salida contiene el resultado de la multiplicación realizada con los elementos pasados a la matriz de entrada.

  > **Important**
  >
  > Asegúrese de sustituir el nombre de recurso de Amazon (ARN) de la función de Lambda en el siguiente código por el ARN de la función que va a crear en el Paso 2.

```
{
  "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"
    }
  }
```

```
}
```

Paso 2: Crear la función de Lambda

En este paso creará la función de Lambda que procesa todos los elementos que se pasan en el lote.

> **Important**
>
> Asegúrese de que su función Lambda esté en la Región de AWS misma posición que su máquina de estados.

Para crear la función de Lambda

1. Use la consola de Lambda para crear una función de Lambda de Python 3.9 denominada **`ProcessEntireBatch`**. Para obtener información sobre la creación de una función de Lambda, consulte el paso 4: Configurar la función de Lambda en el tutorial Cómo empezar a usar el estado Distributed Map.
2. Copie el siguiente código para la función de Lambda y péguelo en la sección Código fuente de la función de Lambda.

```
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. Una vez que haya creado la función de Lambda, copie el ARN de la función que se muestra en la esquina superior derecha de la página. Para copiar el ARN, haga clic en  .

   El siguiente es un ejemplo de ARN, donde *`function-name`* es el nombre de la función de Lambda (en este caso, `ProcessEntireBatch`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Deberá proporcionar la función ARN en la máquina de estado creada en el Paso 1.

4. Elija Implementar para implementar estos cambios.

Paso 3: Ejecutar la máquina de estado

Al ejecutar la máquina de estado, el estado Distributed Map inicia cuatro ejecuciones de flujo de trabajo secundarias, en las que cada ejecución procesa tres elementos, mientras que una ejecución procesa un solo elemento.

En el siguiente ejemplo, se muestran los datos transferidos a la función `ProcessEntireBatch` por una de las ejecuciones del flujo de trabajo secundario.

```
{
  "BatchInput": {
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

Con esta entrada, el siguiente ejemplo muestra la matriz de salida llamada `multiplied` que devuelve la función de Lambda.

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

La máquina de estado devuelve el siguiente resultado, que contiene cuatro matrices denominadas `multiplied` para las cuatro ejecuciones del flujo de trabajo secundario. Estas matrices contienen los resultados de la multiplicación de los elementos de entrada individuales.

```
[
  {
    "statusCode": 200,
    "multiplied": [7, 14, 21]
  },
  {
```

```
    "statusCode": 200,
    "multiplied": [28, 35, 42]
  },
  {
    "statusCode": 200,
    "multiplied": [49, 56, 63]
  },
  {
    "statusCode": 200,
    "multiplied": [70]
  }
]
```

Para combinar todos los elementos de la matriz devueltos en una sola matriz de salida, puede usar el campo ResultSelector. Defina este campo dentro del estado Map Distributed para buscar todas las matrices `multiplied`, extraer todos los elementos de estas matrices y, a continuación, combinarlos en una sola matriz de salida.

Para utilizar el campo `ResultSelector`, actualice la definición de la máquina de estado como se muestra en el siguiente ejemplo.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

La máquina de estado actualizado devuelve una matriz de salida consolidada, como se muestra en el siguiente ejemplo.

```
{
```

```
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Procesamiento de elementos de datos individuales con una función de Lambda

En este tutorial, utilizará el campo ItemBatcher de estado Distributed Map para recorrer en iteración los elementos individuales presentes en un lote mediante una función de Lambda. El estado Distributed Map inicia cuatro ejecuciones de flujos de trabajo secundarios. Cada uno de estos flujos de trabajo secundarios ejecuta un estado Inline Map. Para cada iteración, el estado Inline Map invoca una función de Lambda y pasa un único elemento del lote a la función. A continuación, la función de Lambda procesa el elemento y devuelve el resultado.

Creará una máquina de estado que realiza la multiplicación de una matriz de números enteros. Supongamos que la matriz de enteros que proporciona como entrada es `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` y el factor de multiplicación es 7. Entonces, la matriz resultante formada después de multiplicar estos enteros por un factor de 7, será `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`.

Temas

- Paso 1: Crear la máquina de estado
- Paso 2: Crear la función de Lambda
- Paso 3: Ejecutar la máquina de estado

Paso 1: Crear la máquina de estado

En este paso, creará el prototipo de flujo de trabajo de la máquina de estado que pasa un único elemento de un lote de elementos a cada invocación de la función de Lambda que cree en el Paso 2.

- Utilice la siguiente definición para crear una máquina de estado mediante la consola de Step Functions. Para obtener información sobre cómo crear una máquina de estado, consulte Paso 1: Crear el prototipo de flujo de trabajo en el tutorial Cómo empezar a usar el estado Distributed Map.

  En esta máquina de estado, se define un estado Distributed Map que acepta una matriz de 10 enteros como entrada y pasa estos elementos de la matriz a las ejecuciones del flujo de trabajo secundario en lotes. Cada ejecución del flujo de trabajo secundario recibe un lote de tres

elementos como entrada y ejecuta un estado Inline Map. Cada iteración del estado Inline Map invoca una función de Lambda y pasa un elemento del lote a la función. A continuación, esta función multiplica el elemento por un factor de 7 y devuelve el resultado.

El resultado de la ejecución de cada flujo de trabajo secundario es una matriz JSON que contiene el resultado de la multiplicación de cada uno de los elementos pasados.

> **Important**
>
> Asegúrese de sustituir el nombre de recurso de Amazon (ARN) de la función de Lambda en el siguiente código por el ARN de la función que va a crear en el Paso 2.

```
{
  "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"
        }
      }
```

```
    }
  }
}
```

Paso 2: Crear la función de Lambda

En este paso, se crea la función de Lambda que procesa cada elemento que se pasa del lote.

> **Important**
>
> Asegúrese de que su función Lambda esté en la Región de AWS misma posición que su máquina de estados.

Para crear la función de Lambda

1. Use la consola de Lambda para crear una función de Lambda de Python 3.9 denominada **`ProcessSingleItem`**. Para obtener información sobre la creación de una función de Lambda, consulte el paso 4: Configurar la función de Lambda en el tutorial Cómo empezar a usar el estado Distributed Map.
2. Copie el siguiente código para la función de Lambda y péguelo en la sección Código fuente de la función de Lambda.

```
import json

def lambda_handler(event, context):

    multiplication_factor = event['MyMultiplicationFactor']
    item = event['MyItem']

    result = multiplication_factor * item

    return {
        'statusCode': 200,
        'multiplied': result
    }
```

3. Una vez que haya creado la función de Lambda, copie el ARN de la función que se muestra en la esquina superior derecha de la página. Para copiar el ARN, haga clic en

El siguiente es un ejemplo de ARN, donde *function-name* es el nombre de la función de Lambda (en este caso, `ProcessSingleItem`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Deberá proporcionar la función ARN en la máquina de estado creada en el Paso 1.

4. Elija Implementar para implementar estos cambios.

Paso 3: Ejecutar la máquina de estado

Al ejecutar la máquina de estado, el estado Distributed Map inicia cuatro ejecuciones de flujo de trabajo secundarias, en las que cada ejecución procesa tres elementos, mientras que una ejecución procesa un solo elemento.

En el siguiente ejemplo se muestran los datos que se pasan a una de las invocaciones de funciones `ProcessSingleItem` dentro de la ejecución de un flujo de trabajo secundario.

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

Con esta entrada, el siguiente ejemplo muestra la salida que devuelve la función de Lambda.

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

A continuación se muestra la matriz JSON de salida para una de las ejecuciones del flujo de trabajo secundario.

```
[
  {
    "statusCode": 200,
    "multiplied": 7
  },
```

```
  {
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
  }
]
```

La máquina de estado devuelve el siguiente resultado, que contiene cuatro matrices para las cuatro ejecuciones del flujo de trabajo secundario. Estas matrices contienen los resultados de la multiplicación de los elementos de entrada individuales.

Por último, la salida de la máquina de estado es una matriz denominada `multiplied` que combina todos los resultados de multiplicación devueltos para las cuatro ejecuciones del flujo de trabajo secundario.

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

Para combinar todos los resultados de multiplicación devueltos por las ejecuciones del flujo de trabajo secundario en una única matriz de salida, puede utilizar el campo ResultSelector. Defina este campo dentro del estado Distributed Map para buscar todos los resultados, extraer los resultados individuales y, a continuación, combinarlos en una única matriz de salida denominada `multiplied`.

Para utilizar el campo `ResultSelector`, actualice la definición de la máquina de estado como se muestra en el siguiente ejemplo.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
```

```
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
  }
}
```

La máquina de estado actualizado devuelve una matriz de salida consolidada, como se muestra en el siguiente ejemplo.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Iniciar ejecución de una máquina de estado en respuesta a eventos de Amazon S3

Puede ejecutar una máquina de estado de AWS Step Functions en respuesta a una regla de Amazon EventBridge.

En este tutorial se muestra cómo configurar una máquina de estado como destino de una regla de Amazon EventBridge. Esta regla iniciará la ejecución de una máquina de estado cuando los archivos se agreguen a un bucket de Amazon Simple Storage Service (Amazon S3).

En una aplicación práctica, se podría lanzar una máquina de estado que realice operaciones en los archivos que se agregan al bucket, como la creación de miniaturas o la ejecución de análisis de Amazon Rekognition en archivos de imagen y de vídeo.

En este tutorial, la ejecución de una máquina de estado de `Helloworld` se inicia cargando un archivo a un bucket de Amazon S3. A continuación, se revisa la entrada de ejemplo de esa ejecución para identificar la información que se incluye en la entrada de la notificación de eventos de Amazon S3 enviada a EventBridge.

Temas

Requisito previo: Creación de una máquina de estado

Para configurar una máquina de estado como destino de Amazon EventBridge es necesario crear la máquina de estado.

- Para crear una máquina de estados básica, utilice el tutorial Creating state machine that uses a Lambda function.
- Si ya tiene una maquina de estado `Helloworld`, continúe en el paso siguiente.

Paso 1: Crear un bucket en Amazon S3

Ahora que tiene una máquina de estado de `Helloworld`, debe crear un bucket de Amazon S3 que almacene sus archivos. En el paso 3 de este tutorial configurará una regla para que cuando se agregue un archivo a este bucket, EventBridge desencadene una ejecución de la máquina de estado.

1. Navegue hasta la consola de Amazon S3 y, a continuación, elija Crear bucket para crear el bucket en el que desea almacenar los archivos y desencadenar una regla de eventos de Amazon S3.
2. Escriba un nombre en Nombre del bucket, como *`username`*`-sfn-tutorial`.

> **Note**
>
> Los nombres de bucket deben ser únicos entre todos los nombres de buckets existentes en todas las regiones de AWS de Amazon S3. Utilice su propio *`nombre de usuario`* para que el nombre sea único. Tiene que crear todos los recursos en la misma región de AWS.

3. Mantenga todas las selecciones predeterminadas de la página y elija Crear bucket.

Paso 2: Habilitar notificación de eventos de Amazon S3 con EventBridge

Después de crear el bucket de Amazon S3, configúrelo para que envíe eventos a EventBridge cuando se produzcan determinados eventos en el bucket de S3, como la carga de archivos.

1. Vaya a la consola de Amazon IVS.
2. En la lista Buckets, seleccione el nombre del bucket para el que desea habilitar eventos.
3. Seleccione Propiedades.
4. Desplácese hacia abajo por la página para ver la sección Notificación de eventos y, a continuación, seleccione Editar en la subsección Amazon EventBridge.
5. En Enviar notificaciones a Amazon EventBridge para todos los eventos de este bucket, elija Activar.
6. Elija Guardar cambios.

> **Note**
>
> Después de habilitar EventBridge, los cambios tardan alrededor de cinco minutos en aplicarse.

Paso 3: Crear una regla de Amazon EventBridge

Cuando tenga una máquina de estado, haya creado el bucket de Amazon S3 y lo haya configurado para enviar notificaciones de eventos a EventBridge, cree una regla de EventBridge.

> **Note**
>
> Debe configurar la regla de EventBridge en la misma región de AWS que el bucket de Amazon S3.

Para crear la regla de

1. Vaya a la consola de Amazon EventBridge y seleccione Crear regla .

> **Tip**
>
> Otra opción consiste, en el panel de navegación de la consola de EventBridge, elegir Reglas en Buses y, a continuación, Crear regla.

2. Escriba un Nombre para la regla (por ejemplo, *`S3Step Functions`*) y, si lo desea, introduzca una Descripción.
3. Para Bus de eventos y Tipo de regla, mantenga las selecciones predeterminadas.
4. Elija Siguiente. Se abrirá la página Crear un patrón de eventos.
5. Desplácese hacia abajo hasta la sección Patrón de eventos y haga lo siguiente:

   a. En Origen del evento, deje la selección predeterminada de eventos de AWS o eventos de socios de EventBridge.
   b. En Service de AWS, seleccione Simple Storage Service (S3).
   c. En Tipo de evento, seleccione Notificación de eventos de Amazon S3.
   d. Seleccione Eventos específicos y, a continuación, Objeto creado.
   e. Elija Bucket(s) específico(s) por nombre y escriba el nombre del bucket que creó en el paso 1 (*`username`*`-sfn-tutorial`) para almacenar los archivos.
   f. Elija Siguiente. Se abrirá la página Seleccionar destinos.

Para crear el destino

1. En Destino 1, mantenga la selección predeterminada de Servicio de AWS.
2. En la lista desplegable Seleccionar un destino, seleccione la Máquina de estado de Step Functions.
3. En la lista Máquina de estado, seleccione la máquina de estado que creó anteriormente (por ejemplo, `Helloworld`).
4. Mantenga todas las selecciones predeterminadas de la página y elija Siguiente. Se abrirá la página Configurar etiquetas.
5. Vuelva a seleccionar Siguiente. Se abrirá la página Revisar y crear.
6. Revise los detalles de la regla y elija Crear regla.

   Se creará la regla y se abrirá la página Reglas, donde aparecen todas las reglas de Amazon EventBridge.

Paso 4: Probar la regla de

Ahora que todo está preparado, pruebe a agregar un archivo al bucket de Amazon S3 y, a continuación, observe la entrada de la ejecución resultante de la máquina de estado.

1. Agregue un archivo al bucket de Amazon S3.

   Vaya a la consola de Amazon S3, seleccione el bucket que ha creado para almacenar archivos (*username*`-sfn-tutorial`) y, a continuación, elija Cargar.
2. Agregue un archivo, por ejemplo *`test.png`*, y elija Cargar.

   Esta acción inicia una ejecución de la máquina de estado, que pasa información de AWS CloudTrail como entrada.
3. Compruebe la ejecución de la máquina de estado.

   Vaya a la consola de Step Functions y seleccione la máquina de estado que ha utilizado en la regla de Amazon EventBridge (`Helloworld`).
4. Seleccione la ejecución más reciente de esa máquina de estado y amplíe la sección Entrada de ejecución.

   Esta entrada incluye información como el nombre del bucket y el nombre del objeto. En un caso de uso real, una máquina de estado puede utilizar esta entrada para realizar acciones con ese objeto.

Ejemplo de entrada de ejecución

En el siguiente ejemplo se muestra una entrada típica en la ejecución de la máquina de estado.

```
{
    "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"
    }
  }
```

Crear una API de Step Functions utilizando API Gateway

Puede utilizar Amazon API Gateway para asociar sus AWS Step Functions API a los métodos de una API de API Gateway. Cuando se envía una solicitud HTTPS a un método de la API, API Gateway invoca acciones de la API de Step Functions.

En este tutorial se muestra cómo crear una API que utilice un recurso y el método `POST` para comunicarse con la acción de la API `StartExecution`. Utilizará la consola AWS Identity and Access Management (IAM) para crear un rol para API Gateway. A continuación, usará la consola de API Gateway para crear una API de API Gateway, crear un recurso y un método y asignar el método a la acción de la API `StartExecution`. Por último, implementará y probará su API.

> **Note**
>
> Aunque Amazon API Gateway puede comenzar una ejecución de Step Functions llamando a `StartExecution`, usted debe llamar a `DescribeExecution` para obtener el resultado.

Temas

- Paso 2: Crear un rol de IAM para API Gateway
- Paso 2: Crear una API de API Gateway
- Paso 3: Probar e implementar la API de API Gateway

Paso 2: Crear un rol de IAM para API Gateway

Antes de crear su API de API Gateway, debe conceder a API Gateway permiso para llamar a las acciones de la API de Step Functions.

Para configurar permisos de API Gateway

1. Inicie sesión en la consola de IAM y elija Roles, Crear rol.
2. En la página Seleccionar entidad de confianza, haga lo siguiente:
   a. Para Tipo de entidad de confianza, mantenga la selección predeterminada de Servicio de AWS.
   b. En la lista desplegable, elija API Gateway en la lista desplegable.
3. Seleccione API Gateway y, a continuación, elija Siguiente.
4. Elija Siguiente en la página Agregar permisos.
5. (Opcional) En la página Asignar nombre, revisar y crear, introduzca los detalles, como el nombre del rol. Por ejemplo, escriba **`APIGatewayToStepFunctions`**.
6. Elija Crear rol.

   El rol de IAM se muestra en la lista de roles.
7. Elija el nombre de su rol y anote el ARN de rol, como se muestra en el siguiente ejemplo.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```

Para asociar una política al rol de IAM

1. En la página Roles, busque su rol (`APIGatewayToStepFunctions`) y, a continuación, selecciónelo.
2. En la pestaña Permisos, elija Agregar permisos y, a continuación, Asociar políticas.
3. En la página Asociar política, busque `AWSStepFunctionsFullAccess`, elija la política y después elija Asociar política.

Paso 2: Crear una API de API Gateway

Después de crear el rol de IAM, puede crear la API de API Gateway personalizada.

Para crear la API

1. Abra la consola de Amazon API Gateway y, a continuación, seleccione Create API.
2. En la página Elija un tipo de API, en el panel API de REST, seleccione Crear.
3. En la página Crear API de REST, selecciona Nueva API y, a continuación, introduce ***StartExecutionAPI*** como nombre de la API.
4. Mantenga el Tipo de punto de conexión de la API como Regional y, a continuación, seleccione Crear API.

Para crear un recurso

1. En la página Recursos de la ***StartExecutionAPI***, selecciona Crear recurso.
2. En la página Crear recurso, escriba **`execution`** como Nombre del recurso y, a continuación, elija Crear recurso.

Para crear un método POST

1. Seleccione el recurso /execution y, a continuación, elija Crear método.
2. En Tipo de método, elija `POST`.
3. En Tipo de integración, seleccione Servicio de AWS .
4. En Región de AWS, elija una región de la lista.

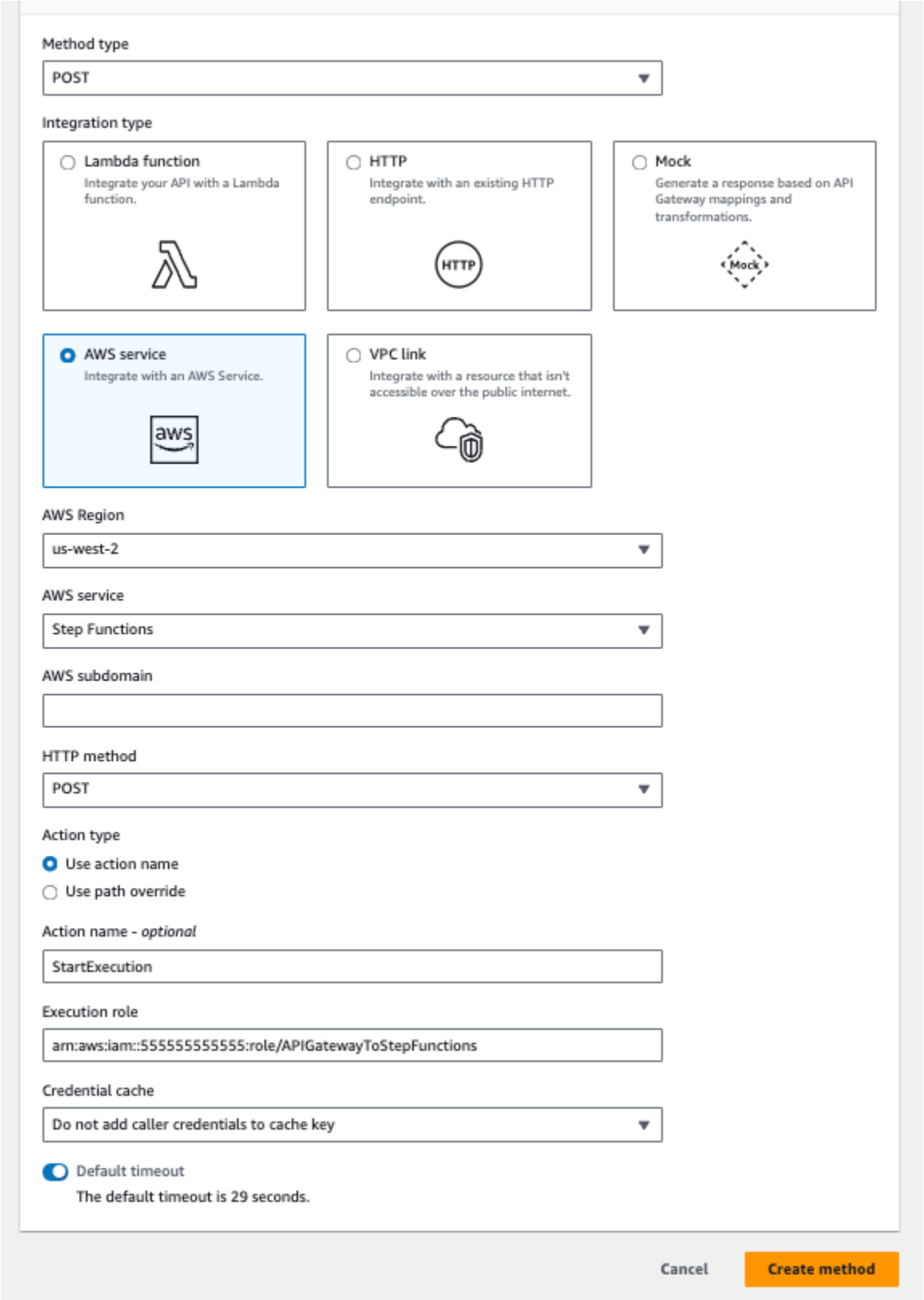
> **Note**
>
> Para conocer las regiones que admiten actualmente Step Functions, consulte Regiones admitidas.

5. En Servicio de AWS, elija Step Functions en la lista.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST en la lista.

> **Note**
>
> Todas las acciones de la API de Step Functions utilizan el método `POST` HTTP.

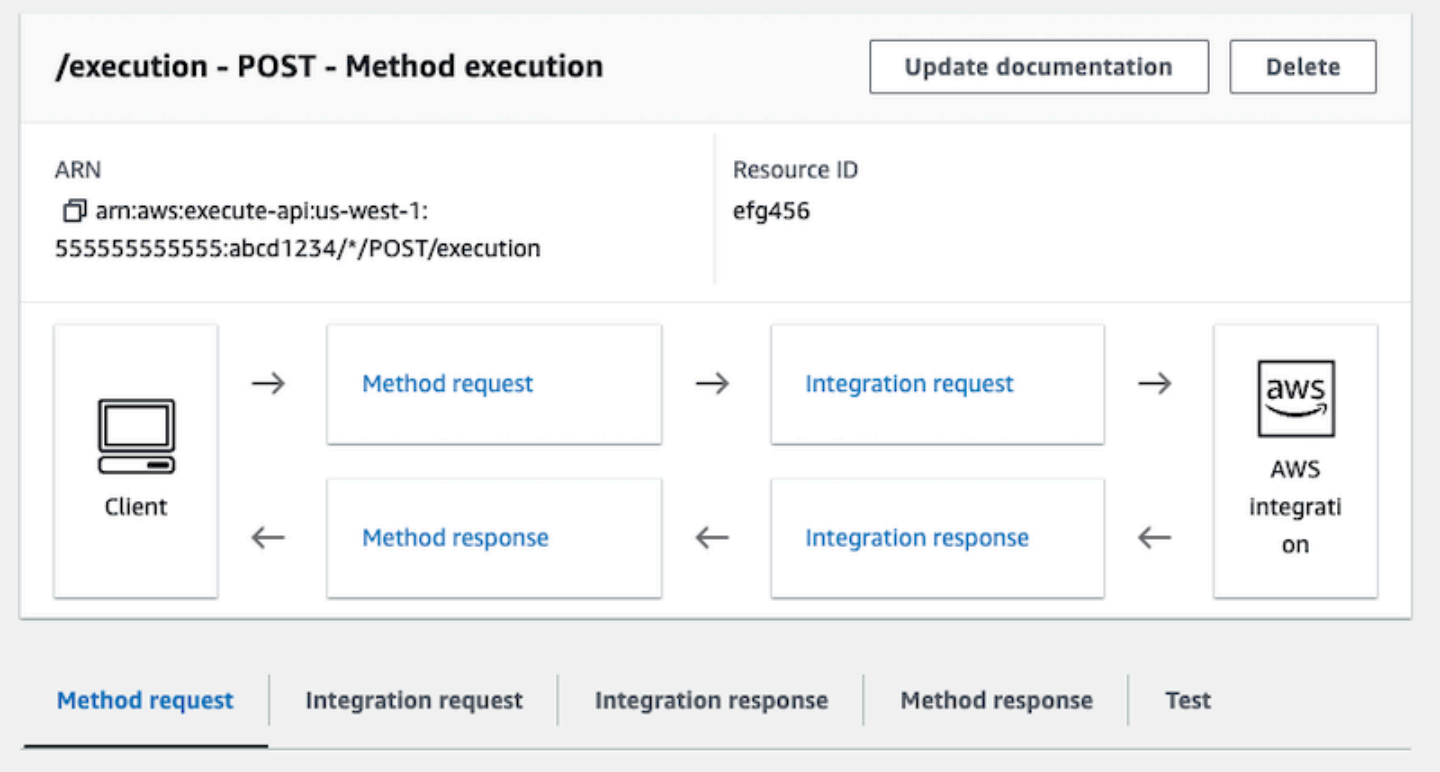
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`StartExecution`**.
10. En Rol de ejecución, escriba el ARN del rol de IAM creado anteriormente, como se muestra en el ejemplo siguiente.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```


Method type
POST
Integration type
Lambda function
Integrate your API with a Lambda function.
HTTP
Integrate with an existing HTTP endpoint.
HTTP
Mock
Generate a response based on API Gateway mappings and transformations.
Mock
AWS service
Integrate with an AWS Service.
aws
VPC link
Integrate with a resource that isn't accessible over the public internet.
AWS Region
us-west-2
AWS service
Step Functions
AWS subdomain
HTTP method
POST
Action type
Use action name
Use path override
Action name - optional
StartExecution
Execution role
arn:aws:iam::555555555555:role/APIGatewayToStepFunctions
Credential cache
Do not add caller credentials to cache key
Default timeout
The default timeout is 29 seconds.
Cancel
Create method

11. Mantenga las opciones predeterminadas para Caché de credenciales y Tiempo de espera predeterminado y, a continuación, seleccione Guardar.

La asignación visual entre API Gateway y Step Functions se muestra en la página /execution - POST - Ejecución de método.



Paso 3: Probar e implementar la API de API Gateway

Una vez que haya creado la API, pruébela e impleméntela.

Para probar la comunicación entre API Gateway y Step Functions

1. En la página /execution - POST - Ejecución de método, elija Probar. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En la pestaña /execution - POST - Prueba de método, copie los siguientes parámetros de solicitud en la sección Cuerpo de la solicitud utilizando el ARN de una máquina de estado existente (o cree una nueva máquina de estado que utilice una función de Lambda) y, a continuación, elija Probar.

```
{
   "input": "{}",
```

```
   "name": "MyExecution",
   "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

Para obtener más información, consulte `StartExecution` Sintaxis de solicitud en la Referencia de la API de AWS Step Functions .

> **Note**
>
> Si no desea incluir el ARN de la máquina de estado en el cuerpo de la llamada a API Gateway, puede configurar una plantilla de asignación en la Solicitud de integración, como se muestra en el ejemplo siguiente.
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> Con este enfoque puede especificar ARN de diferentes máquinas de estado en función de la etapa de desarrollo (por ejemplo, `dev`, `test` y `prod`). Para obtener más información sobre la especificación de variables de etapa en una plantilla de asignación, consulte `$stageVariables` en la Guía para desarrolladores de API Gateway.

3. La ejecución se inicia, y el ARN de ejecución y su fecha de inicio se muestran en Cuerpo de la respuesta.

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> Puede ver la ejecución eligiendo su máquina de estado en la consola de AWS Step Functions.

Para implementar su API

1. En la página de recursos de la ***StartExecutionAPI***, selecciona Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **alpha**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Seleccione Implementar.

Para probar la implementación

1. En la página Etapas de la ***StartExecutionAPI***, expanda alpha,/, /execution, POST y, a continuación, elija el método POST.
2. En Anulaciones de métodos, elija el icono de copia para copiar la URL de invocación de la API. La URL completa deberá ser similar al siguiente ejemplo.

```
https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

3. Desde la línea de comandos, ejecute el comando `curl` utilizando el ARN de la máquina de estado y, a continuación, invoque la dirección URL de su implementación, como se muestra en el siguiente ejemplo.

```
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
```

Se devuelven el ARN de ejecución y su fecha de época, como se muestra en el siguiente ejemplo.

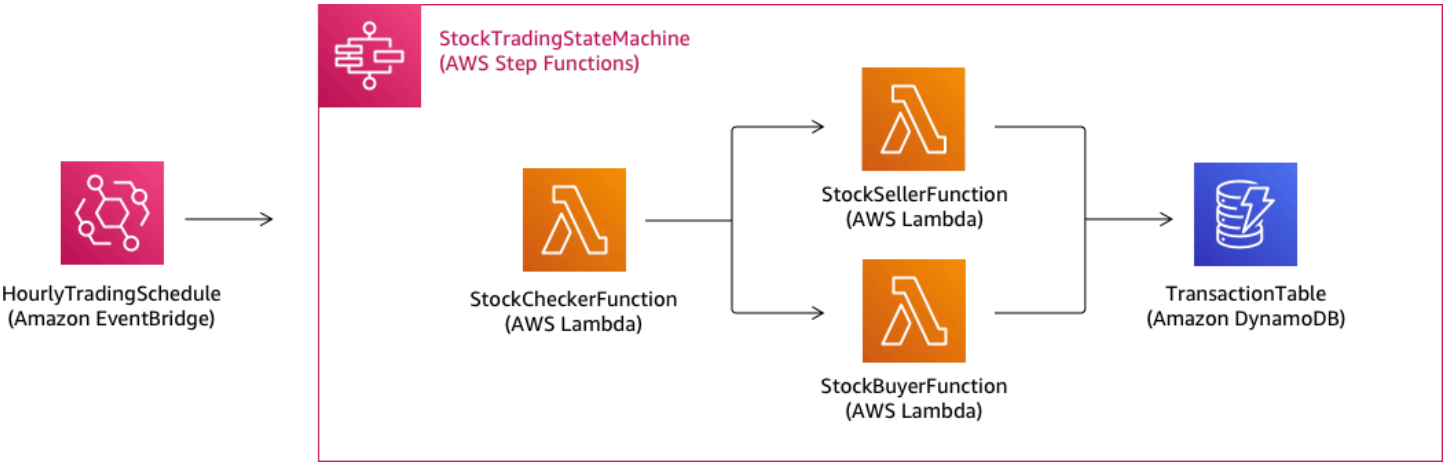
```
{"executionArn":"arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
```

> **Note**
>
> Si aparece el error "Falta el token de autenticación", asegúrese de que la URL de invocación termine en /execution.

Crear una máquina de estado de Step Functions con AWS SAM

En esta guía, descargará, compilará e implementará una aplicación de ejemplo de AWS SAM que contenga una máquina de estado de AWS Step Functions. Esta aplicación crea un flujo de trabajo bursátil simulado que se ejecuta en un horario predefinido (tenga en cuenta que el horario está desactivado de forma predeterminada para evitar incurrir en gastos).

El siguiente diagrama muestra los componentes de esta aplicación:



A continuación se muestra una vista previa de los comandos que se ejecutan para crear la aplicación de ejemplo. Para obtener más detalles sobre cada uno de estos comandos, consulte las secciones más adelante en esta página

```
# 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
```

Requisitos previos

En esta guía se supone que ha completado los pasos de Instalación de la CLI de AWS SAM para su sistema operativo. Se supone que ha hecho lo siguiente:

1. Ha creado una cuenta de AWS.
2. ha configurado los permisos de IAM.
3. Ha instalado Homebrew. Nota: Homebrew solo es un requisito previo para Linux y macOS.
4. Ha instalado la CLI de AWS SAM. Nota: Asegúrese de tener la versión 0.52.0 o posterior. Puede comprobar qué versión tiene ejecutando el comando `sam --version`.

Paso 1: Descargar una aplicación de ejemplo de AWS SAM

Comando a ejecutar:

```
sam init
```

Siga las indicaciones que aparecen en pantalla para seleccionar lo siguiente:

1. Plantilla: plantillas de inicio rápido de AWS
2. Idioma: Python, Ruby, NodeJS, Go, Java o .NET
3. Nombre del proyecto: (nombre que elija, el valor predeterminado es `sam-app`)
4. Aplicación de inicio rápido: flujo de trabajo de varios pasos

Qué está haciendo AWS SAM:

Este comando crea un directorio con el nombre que proporcionó para 'Nombre del proyecto' (el valor predeterminado es `sam-app`). El contenido específico del directorio dependerá del idioma que elija.

A continuación se presentan los contenidos del directorio cuando se elige uno de los tiempos de ejecución de Python:

```
### 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
```

Hay dos archivos especialmente interesantes que puede consultar:

- `template.yaml`: contiene la plantilla de AWS SAM que define los recursos de AWS de su aplicación.
- `statemachine/stockTrader.asl.json`: contiene la definición de máquina de estado de la aplicación, que está escrita en Lenguaje de estados de Amazon.

Puede ver la siguiente entrada en el archivo `template.yaml`, que apunta al archivo de definición de máquina de estado:

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

Puede resultar útil mantener la definición de la máquina de estado como un archivo independiente en lugar de incrustarla en la plantilla de AWS SAM. Por ejemplo, el seguimiento de los cambios en la definición de la máquina de estado es más fácil si no se incluye la definición en la plantilla. Puede usar Workflow Studio para crear y mantener la definición de la máquina de estado y exportar la definición desde la consola directamente al archivo de especificaciones de Amazon States Language sin fusionarla con la plantilla.

Para obtener más información acerca de la aplicación de ejemplo, vea el archivo `README.md` en el directorio del proyecto.

Paso 2: Cree su aplicación

Comando a ejecutar:

Primero cambie al directorio del proyecto (es decir, el directorio donde se encuentra el archivo `template.yaml` de la aplicación de ejemplo; de forma predeterminada es `sam-app`), después, ejecute este comando:

```
sam build
```

Ejemplo de resultados:

```
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
```

Qué está haciendo AWS SAM:

La CLI de AWS SAM viene con abstracciones para una serie de tiempos de ejecución de Lambda para crear sus dependencias y copia todos los artefactos de compilación en carpetas provisionales para que todo esté listo para empaquetarse e implementarse. El comando `sam build` genera todas las dependencias que tiene la aplicación y copia los artefactos de compilación en carpetas bajo `.aws-sam/build`.

Paso 3: Implementar la aplicación en la nube de AWS

Comando a ejecutar:

```
sam deploy --guided
```

Siga las indicaciones que aparecen en pantalla. Puede responder con `Enter` para aceptar las opciones predeterminadas proporcionadas en la experiencia interactiva.

Qué está haciendo AWS SAM:

Este comando implementa su aplicación en la nube de AWS. Toma los artefactos de implementación que crea con el comando `sam build`, los empaqueta y los carga en un bucket de Amazon S3 creado por la CLI de AWS SAM e implementa la aplicación mediante AWS CloudFormation. En la salida del comando deploy (implementar) puede ver los cambios que se están realizando en su pila de AWS CloudFormation.

Puede comprobar que la máquina de estado de ejemplo de Step Functions se implementó correctamente siguiendo estos pasos:

1. Para comenzar, inicie sesión en AWS Management Console y abra la consola de Step Functions en https://console.aws.amazon.com/states/.
2. En el panel de navegación izquierdo, elija Máquinas de estado.
3. Busque y elija su nueva máquina de estado en la lista. Se llamará StockTradingStateMachine-*<unique-hash>*.
4. Elija la pestaña Definición.

Ahora debería ver una representación visual de su máquina de estado. Puede comprobar que la representación visual coincide con la definición de máquina de estado que se encuentra en el archivo `statemachine/stockTrader.asl.json` del directorio del proyecto.

Solución de problemas

Error de la CLI de SAM: "no existe la opción: --guided"

Al ejecutar `sam deploy`, verá el siguiente error:

```
Error: no such option: --guided
```

Esto significa que está utilizando una versión anterior de la CLI de AWS SAM que no admite el parámetro `--guided`. Para solucionar esto, puede actualizar su versión de la CLI de AWS SAM a 0.33.0 o posterior, u omitir el parámetro `--guided` del comando `sam deploy`.

Error de la CLI de SAM: «Error al crear recursos administrados: no se pueden localizar las credenciales»

Al ejecutar `sam deploy`, verá el siguiente error:

```
Error: Failed to create managed resources: Unable to locate credentials
```

Esto significa que no ha configurado las credenciales de AWS para permitir que la CLI de AWS SAM realice llamadas a servicios de AWS. Para solucionarlo, debe configurar las credenciales de AWS. Para obtener más información, consulte Setting Up AWS Credentials in the AWS Serverless Application Model Developer Guide.

Eliminación

Si ya no necesita los recursos de AWS que creó ejecutando este tutorial, puede eliminarlos eliminando la pila de AWS CloudFormation que implementó.

Para eliminar la pila de AWS CloudFormation creada con este tutorial utilizando la AWS Management Console, siga estos pasos:

1. Inicie sesión en la AWS Management Console y abra la consola de AWS CloudFormation en https://console.aws.amazon.com/cloudformation.
2. En el panel de navegación izquierdo, elija Pilas.
3. En la lista de pilas, elija sam-app (o el nombre de la pila que creó).
4. Elija Eliminar.

Cuando termine, el estado de la pila cambiará a DELETE_COMPLETE.

Alternativamente, puede eliminar la pila de AWS CloudFormation ejecutando el siguiente comando de la AWS CLI:

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

Verify Deleted Stack

Para ambos métodos de eliminación de la pila de AWS CloudFormation, puede verificar que se eliminó yendo a https://console.aws.amazon.com/cloudformation, eligiendo Pilas en el panel de

navegación izquierdo y eligiendo Eliminadas en el menú desplegable situado a la derecha del cuadro de texto de búsqueda. Debería ver el nombre de su pila sam-app (o el nombre de la pila que creó) en la lista de pilas eliminadas.

Crear una máquina de estado de actividad con Step Functions

En este tutorial, se explica cómo crear una máquina de estado basada en actividades utilizando Java y AWS Step Functions. Las actividades le permiten controlar código de proceso de trabajo que se ejecuta en otra parte de la máquina de estado. Para obtener información general, consulte Actividades en Cómo funciona Step Functions.

Necesitará lo siguiente para completar este tutorial:

- SDK de para Java. La actividad de ejemplo de este tutorial es una aplicación Java que utiliza el AWS SDK for Java para comunicarse con AWS.
- AWS credenciales en el entorno o en el archivo AWS de configuración estándar. Para obtener más información, consulte Configurar sus AWS credenciales en la Guía para AWS SDK for Java desarrolladores.

Temas

Paso 1: Crear una actividad

Tiene que conseguir que Step Functions reconozca la actividad cuyo proceso de trabajo (programa) desea crear. Step Functions responde con un nombre de recurso de Amazon (ARN) que establece la identidad de la actividad. Debe usar esta identidad para coordinar la información que se pasa entre la máquina de estado y el proceso de trabajo.

> **Important**
>
> Asegúrese de que la tarea de su actividad esté en la misma AWS cuenta que su máquina de estado.

1. En la consola de Step Functions, en el panel de navegación de la izquierda, elija Actividades.
2. Seleccione Crear actividad.
3. Escriba un Nombre de la actividad, por ejemplo, *`get-greeting`* y, a continuación, elija Crear actividad.
4. Cuando se crea la tarea de actividad, tome nota de su ARN, como se muestra en el siguiente ejemplo.

   ```
   arn:aws:states:us-east-1:123456789012:activity:get-greeting
   ```

Paso 2: Crear una máquina de estado

Cree una máquina de estado que determine cuándo se va a invocar la actividad y cuándo el proceso de trabajo debe realizar el trabajo principal, recopilar los resultados y devolverlos. Para crear la máquina de estado, utilice el Editor de código de Workflow Studio.

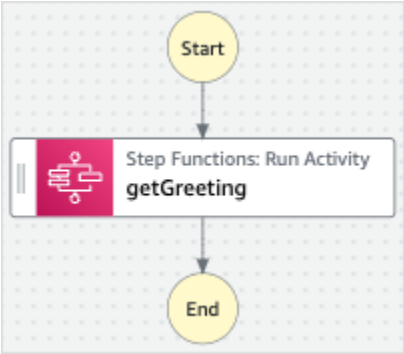
1. En la consola de Step Functions, en el panel de navegación de la izquierda, elija Máquinas de estado.
2. En la página Máquinas de estado, elija Crear máquina de estado.
3. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
4. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
5. Para este tutorial, escribirá la definición de Lenguaje de estados de Amazon (ASL) de su máquina de estado en el editor de código. Para ello, elija Código.
6. Elimine el código reutilizable existente y pegue el siguiente código. Recuerde reemplazar el ARN de ejemplo en este código por el ARN de la tarea de actividad que creó anteriormente en el campo `Resource`.

   ```
   {
     "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
    }
  }
}
```

Esta es una descripción de la máquina de estado realizada mediante el Lenguaje de estados de Amazon (ASL). Define un único estado `Task` llamado `getGreeting`. Para obtener más información, consulte Estructura de las máquinas de estado.

7. En el Panel de visualización de gráficos, asegúrese de que el gráfico de flujo de trabajo de la definición de ASL que ha agregado tiene un aspecto similar al siguiente gráfico.



8. Especifique un nombre para la máquina de estado. Para ello, selecciona el icono de edición situado junto al nombre predeterminado de la máquina de estado MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`ActivityStateMachine`**.

9. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

   Si ya ha creado un rol de IAM con los permisos correctos para su máquina de estado y desea utilizarlo, en Permisos, seleccione Elegir un rol existente y, a continuación, seleccione un rol de

la lista. O seleccione Escribir un ARN de rol y, a continuación, proporcione un ARN para ese rol de IAM.

10. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

    También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 3: Implementar un proceso de trabajo

Cree un proceso de trabajo. Un proceso de trabajo es un programa que se encarga de:

- Sondear Step Functions en busca de actividades utilizando la acción de API `GetActivityTask`.
- Realizar el trabajo de la actividad a través del código especificado (por ejemplo, el método `getGreeting()` del código siguiente).
- Devolver los resultados a través de las acciones de la API `SendTaskSuccess`, `SendTaskFailure` y `SendTaskHeartbeat`.

> **Note**
>
> Para ver un ejemplo de un proceso de trabajo de actividad más completo, consulte Ejemplo de proceso de trabajo de actividad en Ruby. Este ejemplo proporciona una implementación basada en prácticas recomendadas que puede utilizar como referencia para crear un proceso de trabajo de actividad. El código implementa un patrón consumidor-productor con un número configurable de subprocesos para sondeadores y procesos de trabajo de actividad.

Para implementar el proceso de trabajo

1. Cree un archivo denominado `GreeterActivities.java`.
2. Agréguele el código siguiente.

```
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**
>
> La clase `EnvironmentVariableCredentialsProvider` de este ejemplo presupone que se han establecido las variables de entorno `AWS_ACCESS_KEY_ID` (o `AWS_ACCESS_KEY`) y `AWS_SECRET_KEY` (o `AWS_SECRET_ACCESS_KEY`). Para obtener más información sobre cómo proporcionar las credenciales necesarias a la fábrica, consulte AWSCredentialsProviderla referencia de la AWS SDK for Java API y la sección Configuración de AWS credenciales y región para el desarrollo de la guía para AWS SDK for Java desarrolladores.
>
> De forma predeterminada, el AWS SDK esperará hasta 50 segundos para recibir los datos del servidor en relación con cualquier operación. La operación `GetActivityTask` es una operación de sondeo largo que esperará hasta 60 segundos a la siguiente tarea disponible. Para evitar recibir un `SocketTimeoutException` error, SocketTimeout establézcalo en 70 segundos.

3. En la lista de parámetros del constructor `GetActivityTaskRequest().withActivityArn()`, sustituya el valor `ACTIVITY_ARN` por el ARN de la tarea de actividad que creó anteriormente.

Paso 4: Ejecutar la máquina de estado

Cuando se inicia la ejecución de la máquina de estado, el proceso de trabajo sondea Step Functions en busca de actividades, realiza su trabajo (utilizando los datos de entrada proporcionados) y devuelve los resultados.

1. En la ***ActivityStateMachine***página, elija Iniciar ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.

2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   b. En el cuadro Entrada, introduzca la siguiente entrada de JSON para ejecutar el flujo de trabajo.

   ```
   {
     "who": "AWS Step Functions"
   }
   ```

   c. Seleccione Iniciar ejecución.

   d. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente.

Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Paso 5: Ejecutar y detener el proceso de trabajo

Para que el proceso de trabajo pueda sondear la máquina de estado en busca de actividades, es necesario ejecutarlo.

1. En la línea de comandos, vaya al directorio en el que creó `GreeterActivities.java`.
2. Para usar el AWS SDK, agrega la ruta completa de los `third-party` directorios `lib` y a las dependencias del archivo de compilación y a tu archivo de Java. `CLASSPATH` Para obtener más información, consulte Descargue y extraiga el SDK (no se recomienda) en la Guía para desarrolladores de AWS SDK for Java .
3. Compile el archivo.

```
$ javac GreeterActivities.java
```

4. Ejecute el archivo .

```
$ java GreeterActivities
```

5. En la consola de Step Functions, vaya a la página Detalles de ejecución.
6. Cuando se complete la ejecución, examine los resultados.
7. Detenga el proceso de trabajo.

Itera un bucle con Lambda

En este tutorial, implementará un patrón de diseño que utiliza una maquina de estado y una función de AWS Lambda para recorrer en iteración un bucle un número específico de veces.

Utilice este patrón de diseño cada vez que necesite realizar un seguimiento del número de bucles en una máquina de estado. Esta implementación puede ayudarle a desglosar tareas grandes o ejecuciones de ejecución prolongada en trozos más pequeños o finalizar una ejecución después de un número específico de eventos. Puede usar una implementación similar para finalizar y reiniciar periódicamente una ejecución prolongada a fin de evitar sobrepasar las cuotas de servicio de AWS Step Functions AWS Lambda, o de otros AWS servicios.

Antes de empezar, consulte el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda para asegurarse de que está familiarizado con el uso conjunto de Lambda y Step Functions.

Paso 1: Crear una función de Lambda para iterar un recuento

Al utilizar una función de Lambda puede realizar un seguimiento del número de iteraciones de un bucle en su máquina de estado. La función de Lambda siguiente recibe valores de entrada de `count`, `index` y `step`. Devuelve estos valores con un valor de `index` actualizado y un valor booleano denominado `continue`. La función de Lambda establece `continue` en `true` si `index` es menor que `count`.

A continuación, la máquina de estado implementa un estado `Choice` que ejecuta una lógica de aplicación si `continue` es `trueo` finaliza si es `false`.

Para crear la función de Lambda

1. Inicie sesión en la consola de Lambda y, a continuación, elija Crear función.
2. En la página Crear función, elija Diseñar desde cero.
3. En la sección Información básica, configure la función de Lambda de la siguiente manera:
   a. En Nombre de la función, introduzca `Iterator`.
   b. En Runtime (Tiempo de ejecución), elija Node.js.
   c. En Cambiar el rol de ejecución predeterminado, elija Crear un nuevo rol con permisos básicos de Lambda.
   d. Elija Crear función.
4. Copie el siguiente código de la función Lambda en el código fuente.

```
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
  })
}
```

Este código acepta los valores de entrada de `count`, `index` y `step`. Incrementa `index` en el valor de `step` y devuelve estos valores y el valor booleano de `continue`. El valor de `continue` es `true` si `index` es menor que `count`.

5. Elija Implementar.

Paso 2: Probar la función de Lambda

Ejecute su función de Lambda con valores numéricos para verla en acción. Puede proporcionar valores de entrada para la función Lambda que imiten una iteración.

Para probar su función de Lambda

1. Seleccione Probar.
2. En el cuadro de diálogo Configurar un evento de prueba, introduzca `TestIterator` en el cuadro Nombre del evento.
3. Sustituya los datos de ejemplo por lo siguiente.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

Estos valores imitan los que procederían de la máquina de estado durante una iteración. La función Lambda incrementará el índice y `true` regresará `continue` cuando el índice sea inferior a. `count` Para esta prueba, el índice ya se ha incrementado a 5. La prueba se incrementará 6 y se `index` establecerá en. `continue` `true`

4. Seleccione Crear.
5. Elija Probar para probar la función Lambda.

   Los resultados de la prueba se muestran en la pestaña Resultados de la ejecución.

6. Para ver los resultados de la ejecución, elija la pestaña Salida.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Si configura 9 y vuelve `index` a realizar la prueba, los `index` incrementos `continue` serán y serán. `10` `false`

Paso 3: Crear una máquina de estado

> **Antes de salir de la consola Lambda...**
>
> Copie el ARN de la función Lambda. Pégala en una nota. Lo necesitará en el siguiente paso.

A continuación, creará una máquina de estados con los siguientes estados:

- `ConfigureCount`— Establece los valores por defecto para `count`, `index`, `step`.
- `Iterator`— Hace referencia a la función Lambda que creó anteriormente y transfiere los valores configurados en. `ConfigureCount`
- `IsCountReached`— Un estado de elección que continúa el ciclo o pasa al `Done` estado, en función del valor devuelto por la `Iterator` función.
- `ExampleWork`— Un resumen del trabajo que hay que hacer. En este ejemplo, el flujo de trabajo tiene un `Pass` estado, pero en una solución real, lo más probable es que utilices un`Task`.
- `Done`— Estado final de su flujo de trabajo.

Para crear la máquina de estados en la consola:

1. Abra la consola de Step Functions y elija Crear una máquina de estado.

> **Important**
>
> La máquina de estados debe estar en la misma AWS cuenta y región que la función Lambda.

2. Seleccione la plantilla en blanco.
3. En el panel Código, pegue el siguiente JSON que define la máquina de estados.

   Para obtener más información acerca del Amazon States Language, consulte Estructura de las máquinas de estado.

```
{
    "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. Sustituya el `Iterator Resource` campo por el ARN de la función `Iterator` Lambda que creó anteriormente.
5. Seleccione Config e introduzca un nombre para su máquina de estado, como*`IterateCount`*.

> **Note**
>
> Los nombres de máquinas de estado, ejecuciones y tareas de actividad no deben superar los 80 caracteres. Estos nombres deben ser exclusivos para su cuenta y AWS región, y no deben contener ninguno de los siguientes elementos:
>
> - Espacios en blanco
> - Caracteres comodín (`? *`)
> - Caracteres entre corchetes (`< > { } [ ]`)
> - Caracteres especiales (`" # % \ ^ | ~ ` $ & , ; : /`)
> - Caracteres de control (`\\u0000` - `\\u001f` o `\\u007f` - `\\u009f`).
>
> Si la máquina de estado es de tipo rápido, puede proporcionar el mismo nombre a varias ejecuciones de la máquina de estado. Step Functions genera un ARN de ejecución único para cada ejecución de una máquina de estado rápida, incluso aunque varias ejecuciones tengan el mismo nombre.

> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

6. En Tipo, acepte el valor predeterminado de Estándar. En Permisos, elija Crear nuevo rol.
7. Seleccione Crear y, a continuación, confirme la creación de los roles.

Paso 4: Iniciar una nueva ejecución

Después de crear la máquina de estado, puede iniciar una ejecución.

1. En la IterateCountpágina, elija Iniciar la ejecución.
2. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

3. Seleccione Iniciar ejecución.

   Se inicia una nueva ejecución de la máquina de estado, mostrando la ejecución en funcionamiento.
   Vista gráfica de máquina de estados, que muestra el estado del iterador en azul para indicar su estado en progreso.

   La ejecución se incrementa por pasos y realiza el seguimiento del contador mediante la función de Lambda. En cada iteración, realiza el trabajo de ejemplo al que se hace referencia en el estado `ExampleWork` de la máquina de estado.

   Cuando el recuento alcanza el número especificado en el estado `ConfigureCount` de la máquina de estado, la ejecución termina las iteraciones y finaliza.

Vista gráfica de la máquina de estados, que muestra el estado del iterador y el estado Listo en verde para indicar que ambos se han realizado correctamente.

Continuar las ejecuciones de flujos de trabajo de ejecución prolongada como una nueva ejecución

AWS Step Functions se ha diseñado para ejecutar flujos de trabajo que tienen una duración y un número de pasos finitos. Las ejecuciones tienen una duración máxima de un año y un máximo de 25.000 eventos (véase Cuotas).

Para evitar alcanzar la cuota máxima de 25 000 entradas en el historial de eventos de ejecución en ejecuciones prolongadas, le recomendamos que inicie una nueva ejecución de flujos de trabajo directamente desde el estado `Task` de una máquina de estado. Esto le permite dividir los flujos de trabajo en máquinas de estado más pequeñas y continuar el trabajo que está realizando en una nueva ejecución. Para iniciar estas ejecuciones de flujos de trabajo, llame a la acción de la API `StartExecution` desde el estado `Task` y transfiera los parámetros necesarios.

Como alternativa, también puede implementar un patrón que utilice una función de Lambda para iniciar una nueva ejecución de la máquina de estado con el fin de dividir el trabajo en curso entre varias ejecuciones de flujos de trabajo.

En este tutorial se muestran ambos métodos para continuar la ejecución de flujos de trabajo sin superar las cuotas de servicio.

Temas

- Utilizar una acción de la API Step Functions para continuar una nueva ejecución (recomendado)
- Uso de una función de Lambda para continuar con una nueva ejecución

Utilizar una acción de la API Step Functions para continuar una nueva ejecución (recomendado)

Step Functions puede iniciar estas ejecuciones de flujos de trabajo llamando a su propia API como servicio integrado. Le recomendamos que utilice este método para evitar superar las cuotas de servicio en el caso de ejecuciones prolongadas.

Paso 1: Crear una máquina de estado de ejecución prolongada

Cree una máquina de estado de ejecución prolongada que desee iniciar desde el estado `Task` de otra máquina de estado. Para este tutorial, utilice la máquina de estado que utiliza una función de Lambda.

> **Note**
>
> Asegúrese de copiar el nombre y el nombre de recurso de Amazon de esta máquina de estado en un archivo de texto para su uso posterior.

Paso 2: Crear una máquina de estado para llamar a la acción de la API Step Functions

Para iniciar ejecuciones de flujo de trabajo desde un estado **`Task`**

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Desde la pestaña Acciones, arrastra la acción de la StartExecutionAPI y suéltala en el estado vacío denominado Arrastrar primero el estado aquí.
5. Elija el StartExecutionestado y haga lo siguiente en la pestaña Configuración de: Modo Diseño

   a. Cambie el nombre del estado a **`Start nested execution`**.

   b. En Tipo de integración, seleccione AWS SDK: nuevo en la lista desplegable.

   c. En Parámetros de API, haga lo siguiente:

      i. En `StateMachineArn`, sustituya el ejemplo de nombre de recurso de Amazon por el ARN de su máquina de estado. Por ejemplo, introduzca el ARN de la máquina de estado que utiliza Lambda.

      ii. En el nodo `Input`, sustituya el texto del marcador de posición existente por el siguiente valor:

      ```
      "Comment": "Starting workflow execution using a Step Functions API action"
      ```

      iii. Asegúrese de que las entradas en Parámetros de API son similares a las siguientes:

      ```
      {
      ```

```
  "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. (Opcional) Seleccione Definición en el panel Inspector para ver la definición de Lenguaje de estados de Amazon (ASL) generada automáticamente del flujo de trabajo.

   > **Tip**
   >
   > También puede ver la definición de ASL en el Editor de código de Workflow Studio. En el editor de código también puede editar la definición de ASL del flujo de trabajo.

7. Especifique un nombre para la máquina de estado. Para ello, seleccione el icono de edición situado junto al nombre predeterminado de la máquina de estado MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`ParentStateMachine`**.

8. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

   Si ya ha creado un rol de IAM con los permisos correctos para su máquina de estado y desea utilizarlo, en Permisos, seleccione Elegir un rol existente y, a continuación, seleccione un rol de la lista. O seleccione Escribir un ARN de rol y, a continuación, proporcione un ARN para ese rol de IAM.

9. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 3: Actualizar la política de IAM

Para asegurarse de que su máquina de estado tiene permisos para iniciar la ejecución de la máquina de estado que utiliza una función de Lambda, debe asociar una política insertada al rol de IAM de la máquina de estado. Para obtener más información, consulte Para integrar una política insertada en la Guía del usuario de IAM.

1. En la ParentStateMachinepágina, elija el ARN del rol de IAM para ir a la página de roles de IAM de su máquina de estado.
2. Asigne el permiso adecuado a la función de IAM ParentStateMachinepara que pueda iniciar la ejecución de otra máquina de estado. Para asignar estos permisos, haga lo siguiente:
   a. En la página Roles de IAM, elija Agregar permisos y, a continuación, Crear política insertada.
   b. En la página Crear política, elija la pestaña JSON.
   c. Reemplace el texto existente por la política siguiente.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
```

```
    ]
}
```

d. Elija Revisar política.

e. Escriba un nombre para la política y elija Crear política.

Paso 4: Ejecutar la máquina de estado

Las ejecuciones de máquinas de estado son instancias en las que se ejecuta un flujo de trabajo para realizar tareas.

1. En la ParentStateMachinepágina, elija Iniciar la ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.

2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   b. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   c. Seleccione Iniciar ejecución.

   d. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente.

Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

3. Abra la LambdaStateMachinepágina y observe una nueva ejecución provocada por el ParentStateMachine.

Uso de una función de Lambda para continuar con una nueva ejecución

Puede crear una máquina de estado que utilice una función de Lambda para iniciar una nueva ejecución antes de que finalice la ejecución actual. Utilizar este enfoque para continuar su trabajo en curso en una nueva ejecución le permite tener una máquina de estado que puede dividir los trabajos grandes en flujos de trabajo más pequeños o tener una máquina de estado que se ejecute de forma indefinida.

Este tutorial se basa en el concepto de utilizar una función de Lambda externa para modificar el flujo de trabajo, que se presentó en el tutorial Itera un bucle con Lambda. Se utiliza la misma función de Lambda (`Iterator`) para recorrer en iteración un bucle un número específico de veces. Además, se crea otra función de Lambda para comenzar una nueva ejecución del flujo de trabajo y para reducir un contador cada vez que comience una nueva ejecución. Al establecer el número de ejecuciones en la entrada, esta máquina de estado finaliza y reinicia una ejecución un número específico de veces.

La máquina de estado que creará implementa los siguientes estados.

Estado	Finalidad
`ConfigureCount`	Un estado `Pass` que configura los valores de `count`, `index` y `step` que utiliza la función de lambda `Iterator` para recorrer las iteraciones de trabajo.
`Iterator`	Un estado `Task` que hace referencia a la función de Lambda `Iterator`.
`IsCountReached`	Un estado `Choice` que usa un valor booleano de la función `Iterator` para decidir si la máquina de estado debe continuar el trabajo de ejemplo o cambiar al estado `ShouldRestart` .
`ExampleWork`	Un estado `Pass` que representa el estado `Task` que realizaría el trabajo en una implementación real.

Estado	Finalidad
`ShouldRestart`	Un estado `Choice` que utiliza el valor `executionCount` para decidir si debe finalizar una ejecución y comenzar otra, o simplemente finalizar.
`Restart`	Un estado `Task` que usa una función de Lambda para comenzar una ejecución nueva de la máquina de estado. Al igual que la función `Iterator`, esta también disminuye un contador. El estado `Restart` transfiere el valor decrementado del recuento a la entrada de la nueva ejecución.

Requisitos previos

Antes de empezar, consulte el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda para asegurarse de que está familiarizado con el uso conjunto de Lambda y Step Functions.

Temas

- Paso 1: Crear una función de Lambda para iterar un recuento
- Paso 2: Crear una función de Lambda Restart para comenzar una nueva ejecución de Step Functions
- Paso 3: Crear una máquina de estado
- Paso 4: Actualizar la política de IAM
- Paso 5: Ejecutar la máquina de estado

Paso 1: Crear una función de Lambda para iterar un recuento

> **Note**
>
> Si ha completado el tutorial Itera un bucle con Lambda, puede omitir este paso y utilizar esa función de Lambda.

En esta sección y en el tutorial Itera un bucle con Lambda se muestra cómo puede usar una función de Lambda para realizar el seguimiento de un contador, por ejemplo, el número de iteraciones de un bucle en su máquina de estado.

La función de Lambda siguiente recibe valores de entrada de `count`, `index` y `step`. Devuelve estos valores con un valor de `index` y un valor booleano denominado `continue` actualizados. La función de Lambda establece `continue` en `true` si `index` es menor que `count`.

A continuación, la máquina de estado implementa un estado `Choice` que ejecuta una lógica de aplicación si `continue` es `true`, o cambia a `ShouldRestart` si `continue` es `false`.

Crear la función de Lambda de iteración

1. Abra la consola Lambda y, a continuación, elija Crear una función.
2. En la página Crear función, elija Diseñar desde cero.
3. En la sección Información básica, configure la función de Lambda de la siguiente manera:

   a. En Nombre de la función, introduzca `Iterator`.

   b. En Tiempo de ejecución, elija Node.js 16.x.

   c. Mantenga todas las selecciones predeterminadas de la página y elija Crear función.

   Una vez creada la función de Lambda, tome nota de su Nombre de recurso de Amazon (ARN) que aparece en la esquina superior derecha de la página, por ejemplo:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

4. Copie el siguiente código de la función de Lambda en la sección Código fuente de la página ***Iterador*** en la consola de Lambda.

```
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
  })
}
```

Este código acepta los valores de entrada de `count`, `index` y `step`. Incrementa `index` en el valor de `step` y devuelve estos valores y el valor booleano de `continue`. El valor de `continue` es `true` si `index` es menor que `count`.

5. Elija Implementar para implementar el código.

Probar la función de Lambda de iteración

Para ver en acción la función `Iterate`, ejecútela con valores numéricos. Puede proporcionar valores de entrada para la función de Lambda que imiten una iteración para ver qué resultado se obtiene con valores de entrada específicos.

Para probar su función de Lambda

1. En el cuadro de diálogo Configurar evento de prueba, elija Crear nuevo evento de prueba y, a continuación, escriba `TestIterator` en Nombre del evento.
2. Sustituya los datos de ejemplo por lo siguiente.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

Estos valores imitan los que procederían de la máquina de estado durante una iteración. La función de Lambda aumenta el índice y devuelve `continue` como `true`. Cuando el índice no sea menor que `count`, devuelve `continue` como `false`. Para esta prueba, el índice ya se ha incrementado a `5`. Los resultados deben incrementar el valor de `index` a 6 y establecer `continue` en `true`.

3. Seleccione Crear.
4. En la página ***`Iterator`*** de la consola Lambda, TestIteratorasegúrese de que aparezca en la lista y, a continuación, seleccione Probar.

Los resultados de la prueba se muestran en la parte superior de la página. Elija Detalles y examine el resultado.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Si establece `index` en 9 para esta prueba, `index` se incrementa a `10` y `continue` tiene el valor `false`.

Paso 2: Crear una función de Lambda Restart para comenzar una nueva ejecución de Step Functions

1. Abra la consola Lambda y, a continuación, elija Crear una función.
2. En la página Crear función, elija Diseñar desde cero.
3. En la sección Información básica, configure la función de Lambda de la siguiente manera:
   a. En Nombre de la función, introduzca `Restart`.
   b. En Tiempo de ejecución, elija Node.js 16.x.
4. Mantenga todas las selecciones predeterminadas de la página y elija Crear función.

   Una vez creada la función de Lambda, tome nota de su Nombre de recurso de Amazon (ARN) que aparece en la esquina superior derecha de la página, por ejemplo:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

5. Copie el siguiente código de la función de Lambda en la sección Código fuente de la página ***Reinicio*** en la consola de Lambda.

   El siguiente código disminuye un contador del número de ejecuciones y comienza una nueva ejecución de la máquina de estado, incluido el valor reducido.

```
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. Elija Implementar para implementar el código.

Paso 3: Crear una máquina de estado

Ahora que ha creado las dos funciones de Lambda, cree una máquina de estado. En esta máquina de estado, los estados `ShouldRestart` y `Restart` son el modo en que divide su trabajo en varias ejecuciones.

Example ShouldRestart Elija el estado

El siguiente extracto muestra el estado `ShouldRestart`[Choice](). Este estado determina si se debe reiniciar o no la ejecución.

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
```

```
  }
],
```

El valor `$.restart.executionCount` se incluye en la entrada de la ejecución inicial. Se reduce en uno cada vez que se llama a la función `Restart` y, a continuación, se incluye en la entrada de cada ejecución posterior.

Example Estado Task de Restart

El siguiente extracto muestra el estado `RestartTask`. Este estado usa la función de Lambda que creó antes para reiniciar la ejecución y para reducir el contador con el fin de realizar el seguimiento del número restante de ejecuciones que se iniciarán.

```
"Restart": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

Para crear la máquina de estado de

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

   > **Important**
   >
   > Asegúrese de que su máquina de estado esté en la misma AWS cuenta y región que las funciones Lambda que creó anteriormente en los pasos 1 y 2.

2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Para este tutorial, escribirá la definición de Lenguaje de estados de Amazon (ASL) de su máquina de estado en el Editor de código. Para ello, elija Código.
5. Elimine el código reutilizable existente y pegue el siguiente código. Recuerde reemplazar los ARN de este código por los ARN de las funciones de Lambda que creó.

   ```
   {
       "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. Especifique un nombre para la máquina de estado. Para ello, elija el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`ContinueAsNew`**.

7. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

   Si ya ha creado un rol de IAM con los permisos correctos para su máquina de estado y desea utilizarlo, en Permisos, seleccione Elegir un rol existente y, a continuación, seleccione un rol de la lista. O seleccione Escribir un ARN de rol y, a continuación, proporcione un ARN para ese rol de IAM.

8. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

   > **Note**
   >
   > Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de

las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

9. Guarde el nombre de recurso de Amazon (ARN) de esta máquina de estado en un archivo de texto. Deberá proporcionar el ARN y, al mismo tiempo, dar permiso a la función de Lambda para iniciar una nueva ejecución de Step Functions.

Paso 4: Actualizar la política de IAM

Para asegurarse de que la función de Lambda tiene permisos para iniciar una nueva ejecución de , asocie una política insertada al rol de IAM que utilice para la función de Lambda `Restart`. Para obtener más información, consulte Para integrar una política insertada en la Guía del usuario de IAM.

> **Note**
>
> Puede actualizar la línea `Resource` en el ejemplo anterior para hacer referencia al ARN de su máquina de estado `ContinueAsNew`. De este modo se restringe la política de modo que solo comience una ejecución de esa máquina de estado específica.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
     }
 ]
}
```

Paso 5: Ejecutar la máquina de estado

Para comenzar una ejecución, proporcione una entrada que incluya el ARN de la máquina de estado y un valor `executionCount` correspondiente al número de veces que debe comenzar una nueva ejecución.

1. En la ContinueAsNewpágina, elija Iniciar ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.

2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   b. En el cuadro Entrada, introduzca la siguiente entrada de JSON para ejecutar el flujo de trabajo.

      ```
      {
        "restart": {
          "StateMachineArn": "arn:aws:states:us-
      east-1:123456789012:stateMachine:ContinueAsNew",
          "executionCount": 4
        }
      }
      ```

   c. Actualice el campo `StateMachineArn` con el ARN de la máquina de estado `ContinueAsNew`.

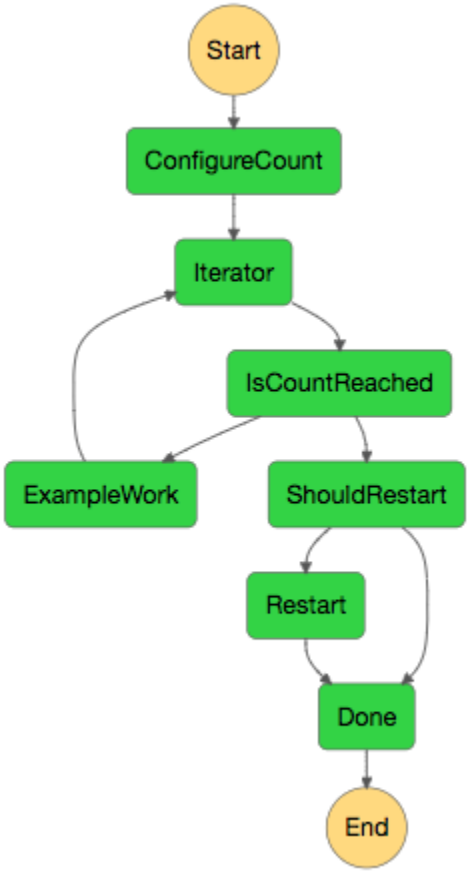
   d. Seleccione Iniciar ejecución.

   e. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente.

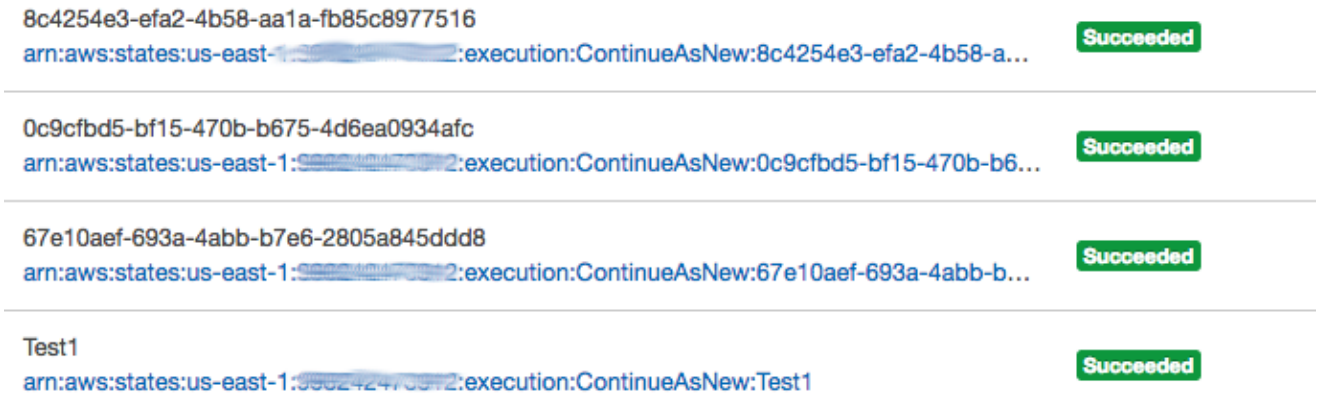
Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

En el gráfico Vista gráfica se muestra la primera de las cuatro ejecuciones. Antes de que finalice, pasará el estado `Restart` y comenzará una nueva ejecución.



Una vez que se completa esta ejecución, puede examinar la siguiente ejecución que se esté ejecutando. Seleccione el ContinueAsNewenlace de la parte superior para ver la lista de ejecuciones. Debe verse la ejecución cerrada recientemente y una ejecución en curso que inició la función de Lambda `Restart`.

Cuando estén completas todas las ejecuciones, debe ver cuatro ejecuciones correctas en la lista. La primera ejecución que se inició muestra el nombre que eligió y las posteriores tienen un nombre generado.



Implementación de un proyecto de aprobación humana de ejemplo

Este tutorial le muestra cómo implementar un proyecto de aprobación humana que le permita a una ejecución de AWS Step Functions detenerse durante una tarea y esperar a que un usuario responda a un correo electrónico. El flujo de trabajo avanza al siguiente estado una vez que el usuario ha aprobado que la tarea continúe.

Al implementar la AWS CloudFormation pila incluida en este tutorial, se crearán todos los recursos necesarios, entre los que se incluyen:

- Recursos de Amazon API Gateway
- Y AWS Lambda funciones
- Una máquina AWS Step Functions de estados
- Un tema de correo electrónico de Amazon Simple Notification Service
- AWS Identity and Access Management Funciones y permisos relacionados

> **Note**
>
> Deberás proporcionar una dirección de correo electrónico válida a la que tengas acceso cuando crees la AWS CloudFormation pila.

Para obtener más información, consulte Trabajar con CloudFormation plantillas y el `AWS::StepFunctions::StateMachine` recurso en la Guía del AWS CloudFormation usuario.

Temas

Paso 1: Crear una AWS CloudFormation plantilla

1. Copie el código de ejemplo de la sección AWS CloudFormation Código fuente de la plantilla.

2. Pegue la fuente de la AWS CloudFormation plantilla en un archivo de su máquina local.

   En este ejemplo, el archivo se llama `human-approval.yaml`.

Paso 2: Crear una pila

1. Inicie sesión en la consola de AWS CloudFormation.
2. Elija Crear pila, y, a continuación, elija Con nuevos recursos (estándar).
3. En la página Crear pila, proceda del modo siguiente:

   a. En la sección Requisitos previos: Preparar plantilla, asegúrese de que esté seleccionada la opción La plantilla está lista.

   b. En la sección Especificar plantilla, elija Cargar un archivo de plantilla y, a continuación, elija Elegir archivo para cargar el archivo `human-approval.yaml` creado anteriormente y que incluye el código fuente de la plantilla.
4. Elija Siguiente.

5. En la página Especificar detalles de pila, haga lo siguiente:

   a. En Nombre de pila, escriba un nombre para la pila.

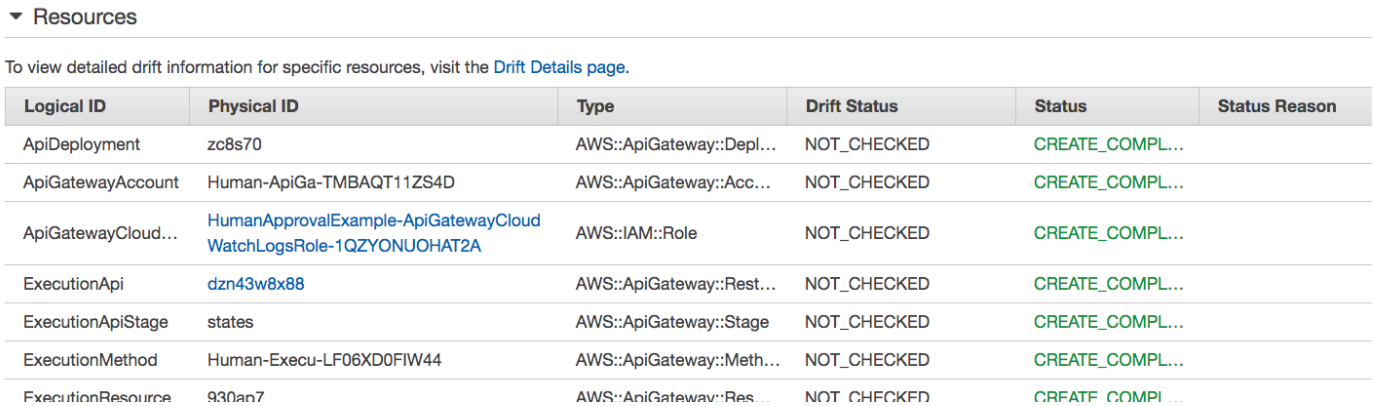
   b. En Parámetros introduzca una dirección de correo electrónico válida. Utilizará esta dirección de correo electrónico para suscribirse al tema de Amazon SNS.

6. Elija Siguiente y después elija Siguiente otra vez.

7. En la página de revisión, elija Acepto que AWS CloudFormation podría crear recursos de IAM y, a continuación, elija Crear.

   AWS CloudFormation comienza a crear su pila y muestra el estado CREATE_IN_PROGRESS. Cuando se complete el proceso, muestra el estado CREATE_COMPLETE. AWS CloudFormation

8. (Opcional) Para mostrar los recursos de la pila, seleccione la pila y elija la pestaña Recursos.



Paso 3: Aprobar la suscripción de Amazon SNS

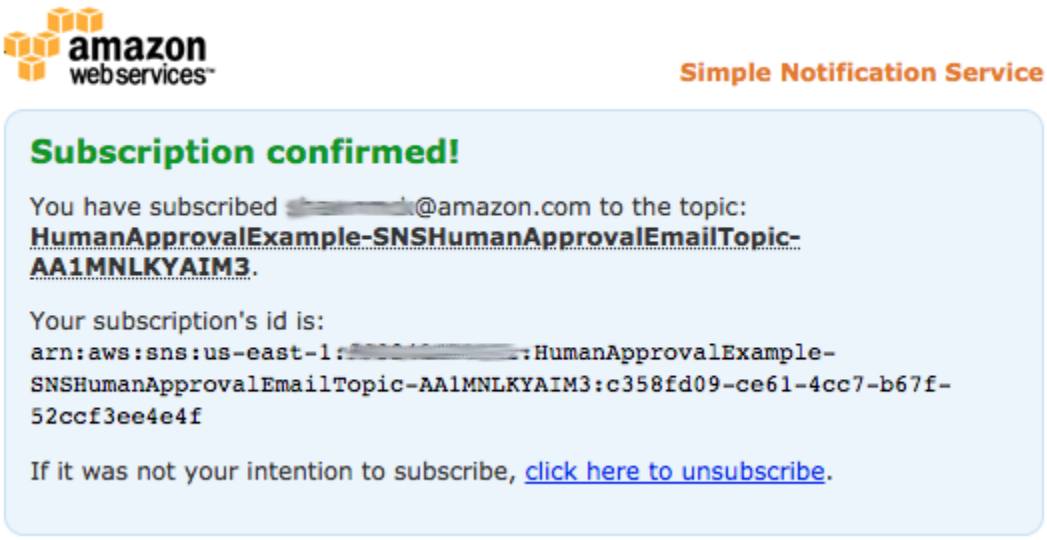
Una vez que se haya creado el tema de Amazon SNS, recibirá un correo electrónico solicitando que confirme la suscripción.

1. Abre la cuenta de correo electrónico que proporcionaste al crear la pila. AWS CloudFormation

2. Abra el mensaje Notificación de AWS : confirmación de suscripción) de parte de no-reply@sns.amazonaws.com

   El correo electrónico incluirá el nombre del recurso de Amazon para el tema de Amazon SNS y un enlace de confirmación.

3. Elija el enlace Confirmar suscripción.



Paso 4: Ejecutar la máquina de estado

1. En la HumanApprovalLambdaStateMachinepágina, selecciona Iniciar la ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.

2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   b. En el cuadro Entrada, introduzca la siguiente entrada de JSON para ejecutar el flujo de trabajo.

```
{
    "Comment": "Testing the human approval tutorial."
```

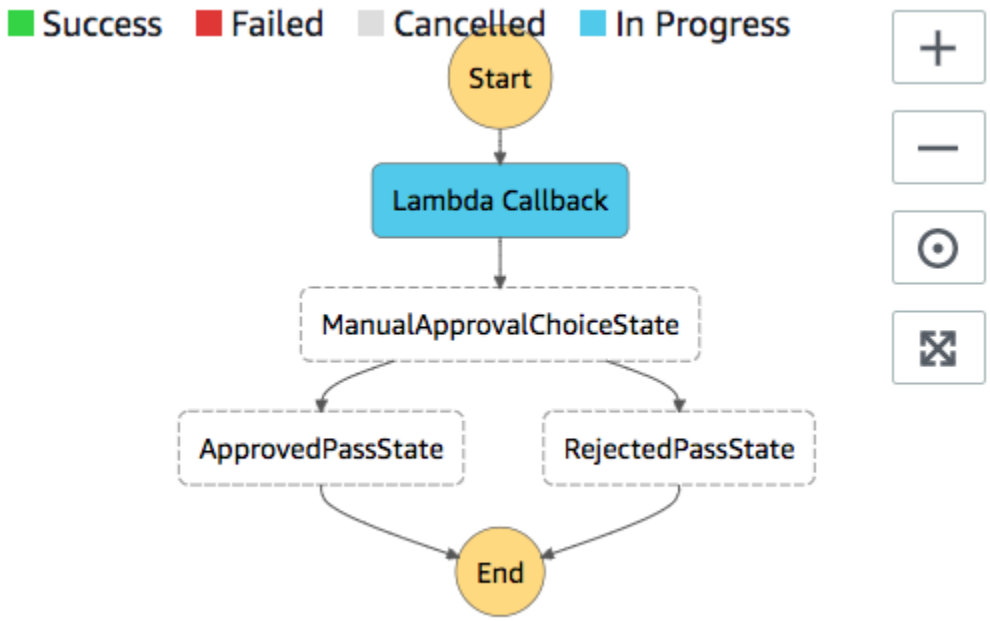
```
}
```

c. Seleccione Iniciar ejecución.

Se inicia la ejecución de la máquina de ApprovalTestestados y se detiene en la tarea Lambda Callback.

d. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

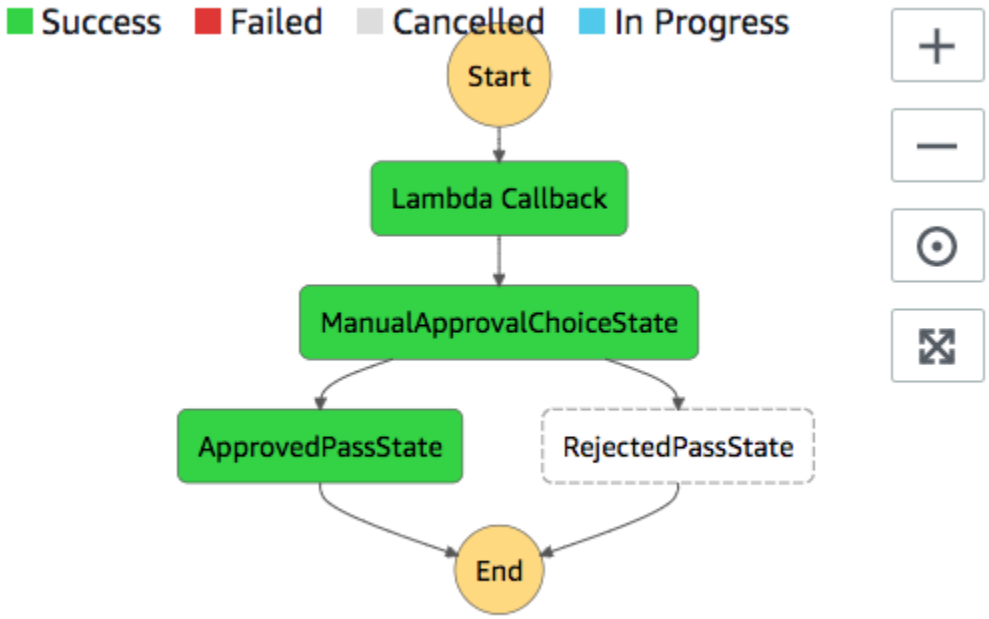


3. En la cuenta de correo electrónico que utilizaste anteriormente para el tema de Amazon SNS, abre el mensaje con el asunto Se requiere la aprobación de. AWS Step Functions

El mensaje incluye URL separadas para Aprobar y Rechazar.

4. Elija la URL Aprobar.

El flujo de trabajo continúa basado en su elección.



AWS CloudFormation Código fuente de la plantilla

Utilice esta AWS CloudFormation plantilla para implementar un ejemplo de flujo de trabajo de un proceso de aprobación humano.

```
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
```

Ver los rastros de X-Ray en Step Functions

En este tutorial, aprenderá a usar X-Ray para rastrear los errores que se producen al ejecutar una máquina de estado. Puede utilizar AWS X-Ray para visualizar los componentes de su máquina de estado, identificar cuellos de botella en el rendimiento y solucionar problemas de solicitudes que dieron lugar a un error. En este tutorial, creará varias funciones de Lambda que producen errores de forma aleatoria, que luego podrá rastrear y analizar mediante X-Ray.

En el tutorial Creación de una máquina de estado de Step Functions que utilice Lambda se explica cómo crear una máquina de estado que llama a una función de Lambda. Si ha completado ese tutorial, vaya al Paso 2 y utilice el rol de (IAM) AWS Identity and Access Management que ha creado anteriormente.

Temas

- paso 1: Crear un rol de IAM para Lambda
- Paso 2: Crear una función de Lambda
- Paso 3: Crear dos funciones de Lambda más
- Paso 4: Crear una máquina de estado
- Paso 5: Ejecutar la máquina de estado

paso 1: Crear un rol de IAM para Lambda

Ambos AWS Lambda AWS Step Functions pueden ejecutar código y acceder a AWS recursos (por ejemplo, datos almacenados en buckets de Amazon S3). Para mantener la seguridad, debe conceder acceso a esos recursos a Lambda y Step Functions.

Lambda requiere que asigne una función AWS Identity and Access Management (de IAM) al crear una función de Lambda, del mismo modo que Step Functions exige que asigne una función de IAM al crear una máquina de estados.

La consola de IAM se utiliza para crear un rol vinculado al servicio.

Para crear un rol (consola)

1. Inicie sesión en la consola de IAM AWS Management Console y ábrala en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola de IAM, elija Roles. A continuación, elija Crear rol.
3. Elija el tipo de rol Servicio de AWS y, a continuación, elija Lambda.
4. Elija el caso de uso Lambda. Los casos de uso son definidos por el servicio de modo tal que ya incluyen la política de confianza exigida por el servicio mismo. A continuación, elija Siguiente: permisos.
5. Elija una o varias políticas de permisos para asociarlas al rol (por ejemplo, `AWSLambdaBasicExecutionRole`). Consulte Modelo de permisos de AWS Lambda.

   Seleccione la casilla situada junto a la política que asigna los permisos que desea que tenga el rol y, a continuación, elija Next: Review.
6. Escriba un Role name.
7. (Opcional) En Descripción del rol, edite la descripción del nuevo rol vinculado al servicio.
8. Revise el rol y, a continuación, seleccione Crear rol.

Paso 2: Crear una función de Lambda

La función de Lambda arrojará errores o se agotará el tiempo de espera de forma aleatoria, lo que generará datos de ejemplo para verlos en X-Ray.

> **Important**
>
> Asegúrese de que su función Lambda esté en la misma AWS cuenta y AWS región que su máquina de estado.

1. Abra la consola de Lambda; y elija Crear función.
2. En la sección Crear función, elija Crear desde cero.
3. En la sección Información básica, configure la función de Lambda:
   a. En Nombre de la función, introduzca `TestFunction1`.
   b. En Tiempo de ejecución, elija Node.js 18.x.
   c. En Rol, seleccione Elegir un rol existente.
   d. En Rol existente, seleccione el rol de Lambda que creó anteriormente.

      > **Note**
      >
      > Si el rol de IAM que creó no aparece en la lista, es posible que necesite unos minutos para que se propague a Lambda.

   e. Seleccione Crear función.

      Una vez creada la función de Lambda, anote su nombre de recurso de Amazon (ARN) en la esquina superior derecha de la página. Por ejemplo:

      ```
      arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
      ```

4. Copie el siguiente código de la función Lambda en la sección Código de función de la página ***`TestFunction1`***.

```
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 }
};
```

Este código crea errores cronometrados aleatoriamente, que se utilizarán para generar errores de ejemplo en su máquina de estados que se pueden ver y analizar mediante trazas de rayos X.

5. Seleccione Guardar.

Paso 3: Crear dos funciones de Lambda más

Crear dos funciones de Lambda mas.

1. Repita el Paso 2 para crear dos funciones de Lambda más. Para la siguiente función, en Nombre de función, introduzca `TestFunction2`. Para la última función, en Nombre de función, introduzca `TestFunction3`.
2. En la consola de Lambda, compruebe que ahora tiene tres funciones de Lambda, `TestFunction1`, `TestFunction2` y `TestFunction3`.

Paso 4: Crear una máquina de estado

En este paso, utilizará la consola de Step Functions para crear una máquina de estado con tres estados `Task`. Cada estado `Task` hará referencia a una de las tres funciones de Lambda.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

> **Important**
>
> Asegúrese de que su máquina de estado esté en la misma AWS cuenta y región que las funciones Lambda que creó anteriormente en los pasos 2 y 3.

2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Para este tutorial, escribirá la definición de Lenguaje de estados de Amazon (ASL) de su máquina de estado en el Editor de código. Para ello, elija Código.

5. Elimine el código reutilizable existente y pegue el siguiente código. En la definición del estado de la tarea, recuerde sustituir los ARN de ejemplo por los ARN de las funciones de Lambda que ha creado.

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

Esta es una descripción de la máquina de estado realizada que utiliza el Amazon States Language. Define tres estados `Task` denominados `CallTestFunction1`, `CallTestFunction2` y `CallTestFunction3`. Cada una de ellas llama a una de las tres funciones de Lambda. Para obtener más información, consulte Estructura de las máquinas de estado.

6. Especifique un nombre para la máquina de estado. Para ello, elija el icono de edición situado junto al nombre de la máquina de estado predeterminada de MyStateMachine. A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`TraceFunctions`**.

7. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Para este tutorial, en Configuración adicional, elija Activar el rastreo de X-Ray. Mantenga el resto de selecciones predeterminadas en la Configuración de máquina de estado.

Si ya ha creado un rol de IAM con los permisos correctos para su máquina de estado y desea utilizarlo, en Permisos, seleccione Elegir un rol existente y, a continuación, seleccione un rol de la lista. O seleccione Escribir un ARN de rol y, a continuación, proporcione un ARN para ese rol de IAM.

8. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

   También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

Paso 5: Ejecutar la máquina de estado

Las ejecuciones de máquinas de estado son instancias en las que se ejecuta un flujo de trabajo para realizar tareas.

1. En la ***TraceFunctions***página, elija Iniciar ejecución.

   Aparece la página Nueva ejecución.

2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que

puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

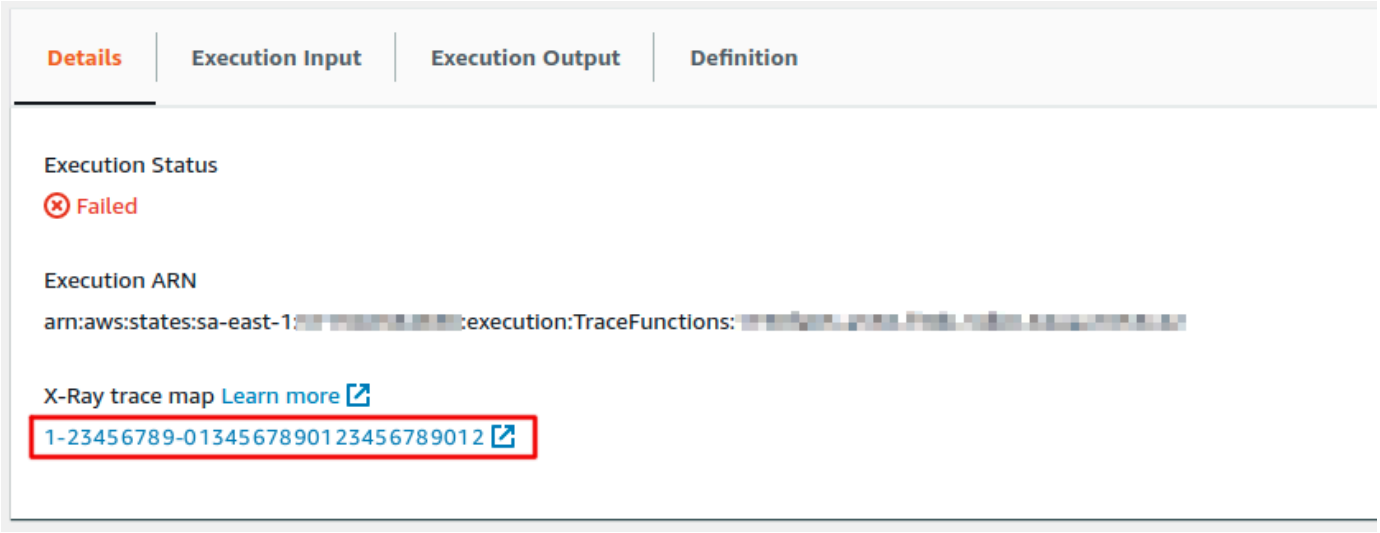
b. Seleccione Iniciar ejecución.

c. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Realice varias ejecuciones (al menos tres).

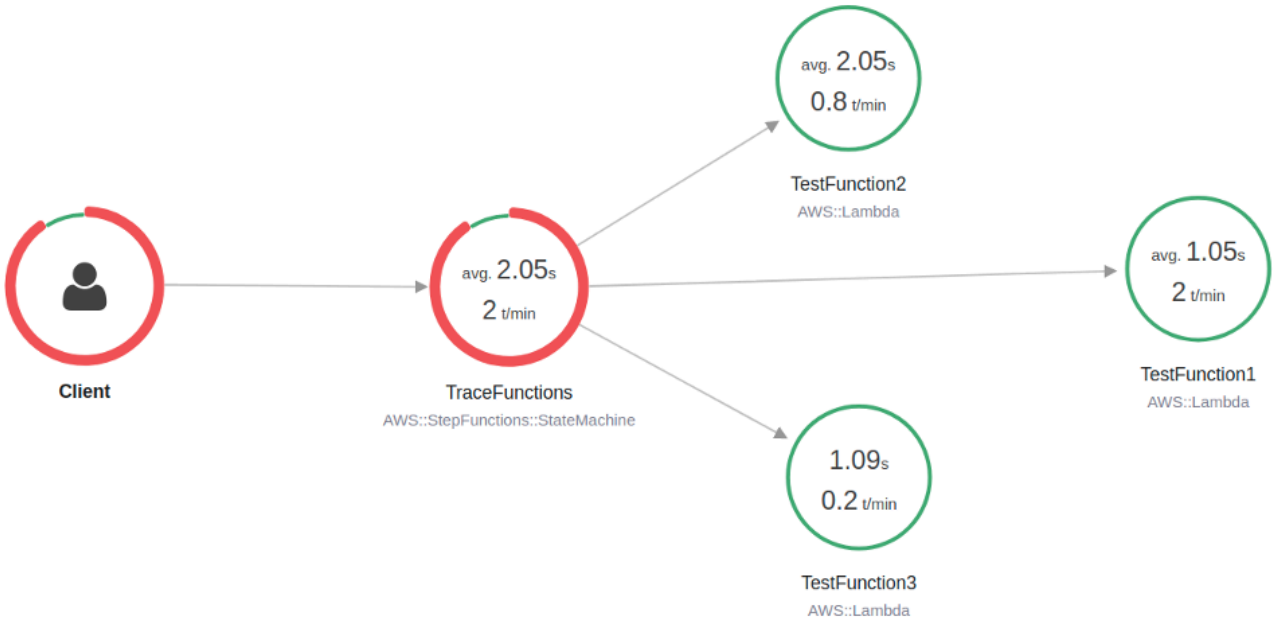
3. Una vez finalizadas las ejecuciones, siga el enlace del mapa de rastreo de X-Ray. Puede ver el rastreo mientras se está ejecutando una ejecución, pero es posible que desee ver los resultados de la ejecución antes de ver el mapa de rastreo de X-Ray.



4. Utilice el mapa de servicio para identificar los servicios donde ocurran errores, donde haya conexiones con alta latencia o rastros de solicitudes que dieron error. En este ejemplo, puede ver cuánto tráfico recibe cada función. Se llamó `TestFunction2` con más frecuencia que `TestFunction3` y se llamó `TestFunction1` con más del doble de frecuencia que `TestFunction2`.

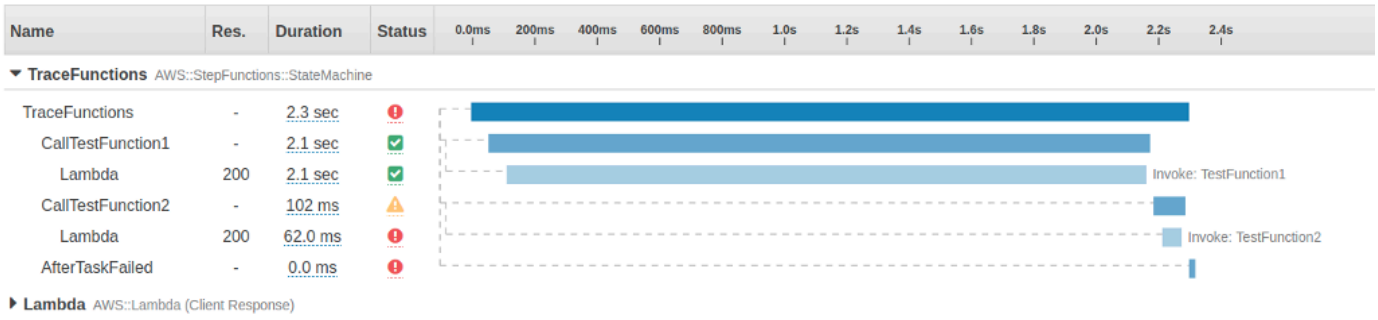
El mapa de servicio indica el estado de cada nodo mediante el uso de colores en función de la proporción de llamadas correctas y errores o fallos:

- El verde se utiliza para las llamadas realizadas con éxito
- El rojo se usa para fallos de servidor (errores de la serie 500)
- El amarillo se usa para los errores del cliente (errores de la serie 400)
- El morado indica los errores de limitación de solicitudes (429: demasiadas solicitudes)



También puede elegir un nodo de servicio para ver sus solicitudes o un límite entre dos nodos para ver las solicitudes que pasaron por esa conexión.

5. Vea el mapa de rastros de X-Ray para ver qué ha sucedido en cada ejecución. La vista Escala de tiempo muestra una jerarquía de segmentos y subsegmentos. La primera entrada de la lista es el segmento, que representa todos los datos registrados por el servicio para una misma solicitud. Debajo del segmento aparecen los subsegmentos. En este ejemplo, se muestran los subsegmentos registrados por las funciones de Lambda.



Para obtener más información sobre cómo entender los rastros de X-Ray y cómo utilizar X-Ray con Step Functions, consulte la AWS X-Ray y Step Functions

Recopile información sobre los buckets de Amazon S3 mediante integraciones de servicios de AWS SDK

En este tutorial se muestra cómo realizar una integración del SDK de AWS con Amazon Simple Storage Service. La máquina de estado que cree en este tutorial recopila información sobre sus buckets de Amazon S3 y, a continuación, enumera los buckets junto con la información de la versión de cada bucket de la región actual.

Temas

- Paso 1: Crear la máquina de estado
- Paso 2: Añadir los permisos de rol de IAM necesarios
- Paso 3: Ejecutar una ejecución estándar de máquina de estado
- Paso 4: Ejecutar una ejecución de máquina de estado rápidos

Paso 1: Crear la máquina de estado

Con la consola de Step Functions, creará una máquina de estado que incluye un estado de `Task` para enumerar todos los buckets de Amazon S3 de la cuenta y la región actuales. A continuación, añadirá otro estado de `Task` que invoque la `HeadBucket` API para comprobar si se puede acceder al bucket devuelto en la región actual. Si no se puede acceder al bucket, la llamada a la API `HeadBucket` devuelve el error `S3.S3Exception`. Incluirá un bloque de `Catch` para capturar esta excepción y un estado de `Pass`como estado alternativo.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

2. En el cuadro de diálogo Elegir una plantilla, seleccione En blanco.
3. Elija Seleccionar. Se abrirá Workflow Studio en Modo Diseño.
4. Para este tutorial, escribirá la definición de Lenguaje de estados de Amazon (ASL) de su máquina de estado en el Editor de código. Para ello, elija Código.
5. Elimine el código reutilizable existente y pegue la siguiente definición de máquina de estado.

```
{
  "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. Especifique un nombre para la máquina de estado. Para ello, elija el icono de edición situado junto al nombre de la máquina de estado predeterminada de. MyStateMachine A continuación, en Configuración de máquina de estado, especifique un nombre en el cuadro Nombre de la máquina de estado.

   En este tutorial, ingrese el nombre **`Gather-S3-Bucket-Info-Standard`**.

7. (Opcional) En Configuración de máquina de estado, especifique otros ajustes del flujo de trabajo, como el tipo de máquina de estado y su función de ejecución.

   Mantenga todas las selecciones predeterminadas en Configuración de máquina de estado.

   Si ya ha creado un rol de IAM con los permisos correctos para su máquina de estado y desea utilizarlo, en Permisos, seleccione Elegir un rol existente y, a continuación, seleccione un rol de la lista. O seleccione Escribir un ARN de rol y, a continuación, proporcione un ARN para ese rol de IAM.

8. En el cuadro de diálogo Confirmar creación de rol, elija Confirmar para continuar.

También puede seleccionar Ver configuración de rol para volver a Configuración de máquina de estado.

> **Note**
>
> Si se elimina el rol de IAM que crea Step Functions, no se podrá volver a crear más adelante. Asimismo, si se modifica el rol (por ejemplo, eliminando Step Functions de las entidades principales de la política de IAM), Step Functions no podrá restablecer la configuración original más adelante.

En el paso 2, añadirá los permisos que faltan al rol de máquina de estado.

Paso 2: Añadir los permisos de rol de IAM necesarios

Para recopilar información sobre los buckets de Amazon S3 en la región actual debe proporcionar a la máquina de estado los permisos necesarios para acceder a los buckets de Amazon S3.

1. En la página de la máquina de estado, elija el ARN del rol de IAM para abrir la página Roles del rol de la máquina de estado.
2. Seleccione Agregar permisos y, a continuación, Crear política insertada.
3. Elija la pestaña JSON y pegue los siguientes permisos en el editor JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Elija Revisar política.
5. En Revisar política, para el Nombre de la política, introduzca **`s3-bucket-permissions`**.
6. Elija Crear política.

Paso 3: Ejecutar una ejecución estándar de máquina de estado

1. En la página Gather-S3-Bucket-Info-Standard, seleccione Iniciar ejecución.
2. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   b. Seleccione Iniciar ejecución.

   c. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Paso 4: Ejecutar una ejecución de máquina de estado rápidos

1. Cree una máquina de estado rápida utilizando la definición de máquina de estado proporcionada en el paso 1. Asegúrese de incluir también los permisos de rol de IAM necesarios, tal como se explica en el paso 2.

   > **Tip**
   >
   > Para diferenciarla de la máquina estándar que creó anteriormente, llame **`Gather-S3-Bucket-Info-Express`** a la máquina de estado rápida.

2. En la página Gather-S3-Bucket-Info-Standard, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   b. Seleccione Iniciar ejecución.
   c. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Herramientas para desarrolladores

Los siguientes recursos contienen información adicional sobre la creación de flujos de trabajo sin servidor y el trabajo con máquinas de estado:

- CDK de AWS

- AWS Kit de herramientas para VS Code

Los temas siguientes contienen información que le enseña a crear, probar y depurar máquinas de estado.

Temas

- Opciones de desarrollo
- AWS Step Functions y AWS SAM
- Uso de Workflow Studio en Application Composer
- Creación de una máquina de estado Lambda para Step Functions mediante AWS CloudFormation
- Creación de una máquina de estado Lambda para Step Functions mediante AWS CDK
- Creación de una API REST de API Gateway con Synchronous Express State Machine mediante AWS CDK
- AWS Step Functions SDK de ciencia de datos para Python
- Implementar máquinas de estado con Terraform

Opciones de desarrollo

Puede implementar sus máquinas AWS Step Functions estatales de varias maneras, por ejemplo, mediante la consola, los SDK o una versión local de Step Functions para las pruebas y el desarrollo.

Temas

- Consola de Step Functions
- AWS SDK
- Flujos de trabajo estándar y rápidos
- API de servicio HTTPS

- Entornos de desarrollo
- puntos de conexión
- AWS CLI
- Step Functions Local
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model y Step Functions
- Terraform y Step Functions
- Compatibilidad con formatos de definición

Consola de Step Functions

Puede definir una máquina de estado mediante la consola de Step Functions. Puede escribir máquinas de estados complejas en la nube sin utilizar un entorno de desarrollo local, ya AWS Lambda que puede utilizar el código para sus tareas. Una vez escrito, puede usar la consola de Step Functions para definir su máquina de estado con Amazon States Language.

En el tutorial Creación de una máquina de estado Lambda se utiliza esta técnica para crear una máquina de estado sencilla, ejecutarla y ver sus resultados.

Simulador de flujo de datos

Puede diseñar, implementar y depurar flujos de trabajo en la consola de Step Functions. También puede controlar el flujo de datos en sus flujos de trabajo mediante el procesamiento de entrada y salida de `JsonPath`. Utilice el simulador de flujo de datos de la consola de Step Functions para descubrir cómo fluye la información de un estado a otro y comprender cómo filtrar y manipular los datos. Esta herramienta simula cada uno de los campos que Step Functions utiliza para procesar datos, como `InputPath`, `Parameters`, `ResultSelector`, `OutputPath` y `ResultPath`.

Para obtener más información, consulte Simulador de flujo de datos.

AWS SDK

Step Functions es compatible con AWS los SDK para Java, .NET, Ruby, PHP, Python (Boto 3) JavaScript, Go y C++. Estos SDK proporcionan una forma cómoda de utilizar las acciones de la API HTTPS de Step Functions en varios lenguajes de programación.

Puede desarrollar máquinas de estado, actividades o iniciadores de máquinas de estado con las acciones de la API expuestas por estas bibliotecas de SDK. También puede obtener acceso a las

operaciones de visibilidad a través de estas bibliotecas para desarrollar sus propias herramientas de monitorización e informes de Step Functions.

Para usar Step Functions con otros AWS servicios, consulta la documentación de referencia de los AWS SDK y herramientas actuales de Amazon Web Services.

> **Note**
>
> Step Functions solo admite puntos de conexión HTTPS.

Flujos de trabajo estándar y rápidos

Al crear una nueva máquina de estado, tiene que seleccionar el `Type` de Estándar o Rápido. En ambos casos, define la máquina de estado mediante Amazon States Language. Las ejecuciones de la máquina de estado se comportarán de forma diferente en función del `Type` que seleccione. El Tipo que seleccione no se podrá cambiar después de que la máquina de estado se haya creado.

Para obtener más información, consulte Registro mediante CloudWatch Logs.

API de servicio HTTPS

Step Functions proporciona operaciones de servicio a las que se puede tener acceso a través de solicitudes HTTPS. Puede utilizar estas operaciones para comunicarse directamente con Step Functions y para desarrollar sus propias bibliotecas en cualquier lenguaje que pueda comunicarse con Step Functions a través de HTTPS.

Puede desarrollar máquinas de estado, procesos de trabajo o iniciadores de máquinas de estado mediante las acciones de la API de servicio. También puede obtener acceso a las operaciones de visibilidad a través de las acciones de la API para desarrollar sus propias herramientas de monitorización e informes.

Para obtener más información sobre las acciones de API, consulte la Referencia de la API de AWS Step Functions.

Entornos de desarrollo

Debe configurar un entorno de desarrollo que se compatible con el lenguaje de programación que vaya a utilizar.

Por ejemplo, para desarrollar para Step Functions con Java, debe instalar un entorno de desarrollo Java, como el AWS SDK for Java, en cada una de sus estaciones de trabajo de desarrollo. Si utiliza el IDE de Eclipse para Java Developers, también debe instalar el AWS Toolkit for Eclipse. Este complemento de Eclipse añade características útiles para el desarrollo en AWS.

Si su lenguaje de programación requiere un entorno de tiempo de ejecución, debe configurar el entorno en cada equipo en el que se ejecutarán estos procesos.

puntos de conexión

Para reducir la latencia y almacenar los datos en una ubicación que cumpla con sus requisitos, Step Functions proporciona puntos de conexión en diferentes AWS regiones.

Cada punto de conexión de Step Functions es totalmente independiente. Una máquina de estado o una actividad solo existe en la región en la que se creó. Ninguna de las máquinas de estado y las actividades que se crean en una región comparte datos ni atributos con las que se crean en otra región. Por ejemplo, puede registrar una máquina de estado con el nombre `STATES-Flows-1` en dos regiones diferentes. La máquina de estado `STATES-Flows-1` de una región no compartirá datos ni atributos con la máquina de estado `STATES-Flow-1` de la otra región.

Para ver una lista de los puntos de conexión de Step Functions, consulte Regiones y puntos de conexión de AWS Step Functions en la Referencia general de AWS.

AWS CLI

Puede acceder a muchas funciones de Step Functions desde AWS Command Line Interface (AWS CLI). AWS CLI Es una alternativa al uso de la consola de Step Functions o, en algunos casos, a la programación mediante las acciones de la API de Step Functions. Por ejemplo, puede utilizar la AWS CLI para crear una máquina de estado y después crear un listado de las máquinas de estado existentes.

Puede usar comandos de Step Functions en la AWS CLI para iniciar y administrar ejecuciones, sondear actividades, registrar latidos de tareas y otras operaciones. Para obtener una lista completa de los comandos de Step Functions, las descripciones de los argumentos disponibles y ejemplos sobre su uso, consulte la Referencia de comandos de AWS CLI .

AWS CLI los comandos siguen de cerca el idioma de Amazon States, por lo que puedes usarlos AWS CLI para obtener información sobre las acciones de la API Step Functions. También puede utilizar sus conocimientos existentes sobre las API para crear un prototipo de código o realizar acciones de Step Functions desde la línea de comandos.

Step Functions Local

Para las pruebas y el desarrollo, puede instalar y ejecutar Step Functions en su equipo local. Con Step Functions Local, puede iniciar una ejecución en cualquier máquina.

La versión local de Step Functions puede invocar AWS Lambda funciones, tanto en AWS ejecución local como cuando se ejecuta localmente. También puede coordinar otros AWS servicios compatibles. Para obtener más información, consulte Probar máquinas de estado de forma local.

> **Note**
>
> Step Functions Local funciona con cuentas ficticias.

AWS Toolkit for Visual Studio Code

Puede utilizar VS Code para interactuar con máquinas de estado remotas y desarrollar máquinas de estado localmente. Puede crear o actualizar máquinas de estado, mostrar las máquinas de estado existentes y ejecutar o descargar una máquina de estado. VS Code también le permite crear nuevas máquinas de estado a partir de plantillas, ver una visualización de su máquina de estado y proporciona fragmentos de código, finalización de código y validación de código.

Para obtener más información, consulte la Guía AWS Toolkit for Visual Studio Code del usuario

AWS Serverless Application Model y Step Functions

Step Functions está integrado con AWS Serverless Application Model, lo que le permite integrar flujos de trabajo con funciones, API y eventos de Lambda para crear aplicaciones sin servidor.

También puede utilizar la AWS SAM CLI junto con la AWS Toolkit for Visual Studio Code como parte de una experiencia integrada.

Para más información, consulte AWS Step Functions y AWS SAM.

Terraform y Step Functions

Terraform by HashiCorp es un marco para crear aplicaciones que utilizan la infraestructura como código (IaC). Con Terraform, puede crear máquinas de estado y utilizar características, como obtener una vista previa de las implementaciones de infraestructura y crear plantillas reutilizables.

Las plantillas de Terraform le ayudan a mantener y reutilizar el código al dividirlo en partes más pequeñas.

Para obtener más información, consulte Implementar máquinas de estado con Terraform.

Compatibilidad con formatos de definición

Step Functions ofrece diversas herramientas con las que puede proporcionar sus definiciones de máquinas de estados en diferentes formatos. Se puede proporcionar una definición de Amazon States Language (ASL) que especifique los detalles de su máquina de estado en forma de cadena o de objeto serializado mediante JSON o YAML.

> **Note**
>
> YAML permite comentarios de una sola línea. Los comentarios de YAML que se proporcionan en la parte de definición de la máquina de estado de una plantilla no se trasladarán a la definición del recurso creado. En su lugar, puede usar la propiedad `Comment` dentro de la definición de la máquina de estado. Para obtener más información, consulte la página Estructura de las máquinas de estado.

En la tabla siguiente se muestran las herramientas que admiten definiciones basadas en ASL.

Compatibilidad con formatos de definición por herramienta

	JSON	YAML	Amazon States Language representado en forma de cadena		
Consola de Step Functions	✓				
API de servicio HTTPS			✓		

	JSON	YAML	Amazon States Language representado en forma de cadena		
AWS CLI			✓		
Step Functions Local			✓		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation y AWS SAM también le permiten cargar sus definiciones de máquina de estados a Amazon S3 en formato JSON o YAML y proporcionar la ubicación de Amazon S3 de la definición en la plantilla. Esto puede mejorar la legibilidad de las plantillas cuando la definición de la máquina de estado sea compleja. Para obtener más información, consulte la página de ubicación AWS::StepFunctions::StateMachine de S3.

Las siguientes AWS CloudFormation plantillas de ejemplo muestran cómo puede proporcionar la misma definición de máquina de estados utilizando diferentes formatos de entrada.

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 y AWS SAM

Puede utilizar la AWS SAM CLI junto con la AWS Toolkit for Visual Studio Code como parte de una experiencia integrada para crear máquinas de estado a nivel local. Puede desarrollar una aplicación sin servidor con AWS SAM y luego construir su máquina de estado en el IDE de VS Code. Posteriormente, puede validar, empaquetar e implementar sus recursos. Si lo desea, también puede publicar en AWS Serverless Application Repository.

> **Tip**
>
> Para implementar un ejemplo de aplicación sin servidor que inicie un flujo de trabajo de Step Functions utilizando AWS SAM su Cuenta de AWS, consulte el Módulo 11: Implementar con AWS SAM The AWS Step Functions Workshop.

Temas

- ¿Por qué usar Step Functions con AWS SAM?

¿Por qué usar Step Functions con AWS SAM?

Cuando utilizas Step Functions con AWS SAM , puedes:

- Comience a usar una plantilla AWS SAM de ejemplo.
- Crear su máquina de estado en su aplicación sin servidor.
- Utilizar la sustitución de variables para sustituir ARN en su máquina de estado en el momento de la implementación.

  AWS CloudFormation admite `DefinitionSubstitutions` que permiten añadir referencias dinámicas en la definición de su flujo de trabajo a un valor que proporcione en la plantilla de CloudFormation. Puede añadir referencias dinámicas añadiendo sustituciones a la definición del flujo de trabajo mediante la notación `${dollar_sign_brace}`. También debe definir estas referencias dinámicas en la `DefinitionSubstitutions` propiedad del StateMachine recurso de la CloudFormation plantilla. Estas sustituciones se sustituyen por valores reales durante el proceso de creación de la pila de CloudFormation. Para obtener más información, consulte DefinitionSubstitutions en plantillas AWS SAM.
- Especifique la función de su máquina de estados mediante plantillas AWS SAM de políticas.
- Inicie las ejecuciones de máquinas de estado con API Gateway, EventBridge eventos o según una programación incluida en su AWS SAM plantilla.

Integración de Step Functions con la especificación de AWS SAM

Puede utilizar las plantillas de políticas de AWS SAM para agregar permisos a su máquina de estado. Con estos permisos, puede orquestar funciones de Lambda y otros recursos de AWS para formar flujos de trabajo complejos y sólidos.

Integración de Step Functions con la CLI de SAM

Step Functions está integrado con la AWS SAM CLI. Utilice esto para desarrollar rápidamente una máquina de estado en su aplicación sin servidor.

Pruebe el Crear una máquina de estado de Step Functions con AWS SAM tutorial para aprender a utilizarlas AWS SAM para crear máquinas de estado.

Las funciones AWS SAM CLI compatibles incluyen:

Comando de la CLI	Descripción
sam init	Inicializa una aplicación sin servidor con una AWS SAM plantilla. Se puede utilizar con una plantilla de SAM para Step Functions.
sam validate	Valida una plantilla. AWS SAM
sam package	Empaqueta una AWS SAM aplicación. Crea un archivo ZIP de su código y dependencias y luego lo carga en Amazon S3. Después, el comando devuelve una copia de la plantilla de AWS SAM , sustituyendo las referencias a artefactos locales con la ubicación de Amazon S3 donde el comando cargó los artefactos.
sam deploy	Implementa una AWS SAM aplicación.
sam publish	Publica una AWS SAM aplicación en. AWS Serverless Application Repository Este comando toma una AWS SAM plantilla empaquetada y publica la aplicación en la región especificada.

> **Note**
>
> Si usa AWS SAM local, puede emular Lambda y API Gateway localmente. Sin embargo, no puedes emular Step Functions localmente usando AWS SAM.

DefinitionSubstitutions en plantillas AWS SAM

Puede definir máquinas de estado utilizando una plantilla de CloudFormation con AWS SAM. Con AWS SAM, puede definir la máquina de estado insertándola en la plantilla o en un archivo independiente. La siguiente plantilla de AWS SAM incluye una máquina de estado que simula un flujo de trabajo de cotización de acciones. Esta máquina de estado invoca tres funciones de Lambda para comprobar el precio de una acción y determinar si se debe comprar o vender la acción. A continuación, esta transacción se registra en una tabla de Amazon DynamoDB. Los ARN de las funciones de Lambda y la tabla DynamoDB de la siguiente plantilla se especifican mediante `DefinitionSubstitutions`.

```
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
```

El siguiente código es la definición de máquina de estado del archivo `stock_trader.asl.json` que se utiliza en el tutorial Crear una máquina de estado de Step Functions con AWS SAM. Esta definición de máquina de estado contiene varias `DefinitionSubstitutions` que se denotan mediante notación `${dollar_sign_brace}`. Por ejemplo, en lugar de especificar un ARN de función de Lambda estática para la tarea `Check Stock Value`, se utiliza la sustitución `${StockCheckerFunctionArn}`. Esta sustitución se define en la propiedad DefinitionSubstitutions de la plantilla. `DefinitionSubstitutions` es un mapa de pares de clave-valor para el recurso de la máquina de estado. En`DefinitionSubstitutions`, $ {StockCheckerFunctionArn} se asigna al ARN del `StockCheckerFunction` recurso mediante la función CloudFormation intrínseca. `!GetAtt` Al implementar la plantilla de AWS SAM, las `DefinitionSubstitutions` de la plantilla se sustituyen por los valores reales.

```
{
    "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
        }
    }
}
```

Siguientes pasos

Puede obtener más información sobre el uso de Step Functions AWS SAM con los siguientes recursos:

- Completa el Crear una máquina de estado de Step Functions con AWS SAM tutorial para crear una máquina de estados con AWS SAM.
- Especifique un AWS::Serverless::StateMachinerecurso.
- Busque las plantillas de políticas de AWS SAM que desea utilizar.
- Use AWS Toolkit for Visual Studio Code con Step Functions.
- Consulte la referencia de la CLI de AWS SAM para obtener más información sobre las características disponibles en AWS SAM.

También puede diseñar y crear sus flujos de trabajo en infraestructura como código (IaC) mediante generadores visuales, como Workflow Studio en Application Composer. Para obtener más información, consulte Uso de Workflow Studio en Application Composer.

Uso de Workflow Studio en Application Composer

AWS Application Composer es un generador visual que le ayuda a desarrollar plantillas de AWS SAM y AWS CloudFormation mediante una interfaz gráfica sencilla. Con Application Composer puede diseñar la arquitectura de aplicaciones mediante arrastrar, agrupar y conectar Servicios de AWS en un lienzo visual. A continuación, Application Composer crea una plantilla de infraestructura como código (IaC) a partir del diseño que puede utilizar para implementar la aplicación con la interfaz de la línea de comandos de AWS SAM (CLI de AWS SAM) o CloudFormation. Para obtener más información sobre Application Composer, consulte ¿Qué es Application Composer?

Workflow Studio está disponible en Application Composer para ayudarle a diseñar y crear sus flujos de trabajo. Workflow Studio en Application Composer proporciona un entorno visual de IaC que facilita la incorporación de flujos de trabajo en sus aplicaciones sin servidor creadas con herramientas de IaC, como plantillas CloudFormation. Al utilizar Workflow Studio en Application Composer, conecta los pasos individuales del flujo de trabajo con recursos de AWS y genera las configuraciones de los recursos en una plantilla de AWS SAM. También añade los permisos de IAM necesarios para que se ejecute el flujo de trabajo. Con Workflow Studio en Application Composer puede crear prototipos de sus aplicaciones y convertirlos en aplicaciones listas para producción.

Al utilizar Workflow Studio en Application Composer puede cambiar entre el lienzo de Application Composer y Workflow Studio.

Temas

- Uso de Workflow Studio en Application Composer para crear un flujo de trabajo sin servidor

- Haga referencia a los recursos de manera dinámica mediante sustituciones de definición de CloudFormation en Workflow Studio
- Conecte tareas de integración de servicios a tarjetas de componentes mejoradas
- Importar proyectos existentes y sincronizarlos localmente
- Características de Workflow Studio no disponibles en AWS Application Composer

Uso de Workflow Studio en Application Composer para crear un flujo de trabajo sin servidor

1. Abra la consola de Application Composer y elija Crear proyecto para crear un proyecto.
2. En el campo de búsqueda de la paleta Recursos, ingrese **`state machine`**.
3. Arrastre el recurso Máquina de estado de Step Functions al lienzo.
4. Elija Editar en Workflow Studio para editar su recurso de máquina de estado.

   La siguiente animación muestra cómo puede cambiar a Workflow Studio para editar la definición de máquina de estado.
   Una animación que ilustra cómo puede utilizar Workflow Studio en Application Composer.

   La integración con Workflow Studio para editar recursos de máquinas de estado creados en Application Composer solo está disponible para el recurso `AWS::Serverless::StateMachine`. Esta integración no está disponible para las plantillas que utilizan el recurso `AWS::StepFunctions::StateMachine`.

Haga referencia a los recursos de manera dinámica mediante sustituciones de definición de CloudFormation en Workflow Studio

En Workflow Studio, puede utilizar sustituciones de definición de CloudFormation en su definición de flujo de trabajo para hacer referencia de manera dinámica a los recursos que ha definido en la plantilla de IaC. Puede añadir sustituciones de marcadores de posición a su definición de flujo de trabajo utilizando la notación `${dollar_sign_brace}` y se sustituirán por valores reales durante el proceso de creación de pila de CloudFormation. Para obtener más información sobre sustituciones de definición, consulte DefinitionSubstitutions en plantillas AWS SAM.

La siguiente animación muestra cómo puede añadir sustituciones de marcadores de posición para los recursos en su definición de máquina de estado.

Una animación que ilustra cómo hacer referencia de manera dinámica a recursos, como funciones AWS Lambda o sustituciones de definición cuando se utiliza Workflow Studio en Application Composer.

Conecte tareas de integración de servicios a tarjetas de componentes mejoradas

Puede conectar las tareas que llaman a integraciones de servicios optimizadas con tarjetas de componentes mejoradas en el lienzo de Application Composer. De este modo, se asignan automáticamente las sustituciones de marcadores de posición especificadas por la notación `${dollar_sign_brace}` en la definición de flujo de trabajo y la propiedad `DefinitionSubstitution` del recurso `StateMachine`. También se agregan las políticas de AWS SAM adecuadas para la máquina de estado.

Si asigna tareas de integración de servicios optimizadas con tarjetas de componentes estándar, la línea de conexión no aparece en el lienzo de Application Composer.

La siguiente animación muestra cómo conectar una tarea optimizada a una tarjeta de componentes mejorada y ver los cambios en Inspector de cambios.
Una animación que ilustra cómo conectar tareas que llaman a integraciones de servicios optimizadas con tarjetas de componentes mejoradas cuando se utiliza Workflow Studio en Application Composer.

No puede conectar integraciones del SDK de AWS en el estado Task con tarjetas de componentes mejoradas ni integraciones de servicios optimizadas con tarjetas de componentes estándar. Para estas tareas, puede asignar las sustituciones en el panel Propiedades de recursos en el lienzo de Application Composer y añadir políticas en la plantilla de AWS SAM.

> **Tip**
>
> Como alternativa, también puede asignar sustituciones de marcadores de posición para su máquina de estado en Sustituciones de definición en el panel Propiedades de recursos. Al hacerlo, debe añadir los permisos necesarios para las llamadas al estado Task de Servicio de AWS en el rol de ejecución de máquina de estado. Para obtener información acerca de los permisos que podría necesitar su rol de ejecución, consulte Roles de ejecución en Workflow Studio.

La siguiente animación muestra cómo puede actualizar manualmente la asignación de sustituciones de marcadores de posición en el panel Propiedades de recursos.

Una animación que ilustra cómo actualizar manualmente la asignación de sustituciones de marcadores de posición en el panel Propiedades de recursos cuando se utiliza Workflow Studio en Application Composer.

Importar proyectos existentes y sincronizarlos localmente

Puede abrir proyectos de CloudFormation y AWS SAM existentes en Application Composer para visualizarlos y comprender mejor sus diseños y modificarlos. Con la característica de sincronización local de Application Composer puede sincronizar y guardar automáticamente sus archivos de plantilla y código en la máquina de compilación local. El uso del modo de sincronización local puede complementar los flujos de desarrollo actuales. Asegúrese de que su navegador es compatible con la API de acceso al sistema de archivos, que permite a las aplicaciones web leer, escribir y guardar archivos en su sistema de archivos local. Recomendamos utilizar Google Chrome o Microsoft Edge.

Características de Workflow Studio no disponibles en AWS Application Composer

Al utilizar Workflow Studio en Application Composer, algunas de las características de Workflow Studio no están disponibles. Además, la sección Parámetros de API disponible en el panel de Inspector admite sustituciones de definición de CloudFormation. Puede añadir las sustituciones en Modo Código utilizando la notación `${dollar_sign_brace}`. Para obtener más información acerca de esta notación, consulte DefinitionSubstitutions en plantillas AWS SAM.

La siguiente lista describe las características de Workflow Studio que no están disponibles cuando se utiliza Workflow Studio en Application Composer:

- Plantillas de inicio: las plantillas de inicio son proyectos de ejemplo listos para ejecutarse que crean automáticamente los prototipos y las definiciones de flujo de trabajo. Estas plantillas implementan todos los recursos de AWS relacionados que su proyecto necesita para Cuenta de AWS.
- Modo de configuración: este modo permite administrar la configuración de sus máquinas de estado. Puede actualizar las configuraciones de sus máquinas de estado en las plantillas de IaC o utilizar el panel Propiedades de recursos en el lienzo de Application Composer. Para obtener información sobre la actualización de configuraciones en el panel Propiedades de recursos, consulte Conecte tareas de integración de servicios a tarjetas de componentes mejoradas.
- API TestState
- Opción para importar o exportar definiciones de flujo de trabajo desde el botón desplegable Acciones de Workflow Studio. En su lugar, en el menú de Application Composer, seleccione

Abrir > Carpeta del proyecto. Asegúrese de que ha activado el modo de sincronización local para guardar automáticamente los cambios en el lienzo de Application Composer directamente en su máquina local.

- Botón Ejecutar. Cuando utiliza Workflow Studio en Application Composer, Application Composer genera el código IaC para el flujo de trabajo. Por tanto, debe implementar primero la plantilla. A continuación, ejecute el flujo de trabajo en la consola o a través de AWS Command Line Interface (AWS CLI).

Creación de una máquina de estado Lambda para Step Functions mediante AWS CloudFormation

En este tutorial se muestra cómo crear una AWS Lambda función básica utilizando AWS CloudFormation. Utilizará la AWS CloudFormation consola y una plantilla YAML para crear la pila (las funciones de IAM, la función Lambda y la máquina de estados). A continuación, utilizará la AWS Step Functions consola para iniciar la ejecución de la máquina de estados.

Para obtener más información, consulte Trabajar con CloudFormation plantillas y el `AWS::StepFunctions::StateMachine` recurso en la Guía del AWS CloudFormation usuario.

Temas

- Paso 1: Configure la AWS CloudFormation plantilla
- Paso 2: Utilice la AWS CloudFormation plantilla para crear una máquina de estado Lambda
- Paso 3: Iniciar la ejecución de una máquina de estado

Paso 1: Configure la AWS CloudFormation plantilla

Antes de utilizar las plantillas de ejemplo, debe comprender cómo declarar las distintas partes de una plantilla de AWS CloudFormation .

Temas

- Para crear un rol de IAM para Lambda
- Para crear una función de Lambda
- Para crear un rol de IAM para la ejecución de la máquina de estado
- Para crear una máquina de estado Lambda

Para crear un rol de IAM para Lambda

Defina la política de confianza asociada al rol de IAM para la función de Lambda. En los siguientes ejemplos se define una política de confianza mediante YAML o JSON.

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"
                      }
                  ]
              }
          }
```

Para crear una función de Lambda

Defina las siguientes propiedades de la función de Lambda que imprimirá el mensaje `Hello World`.

> **Important**
>
> Asegúrese de que su función Lambda esté en la misma AWS cuenta y AWS región que su máquina de estado.

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"
            }
        },
```

Para crear un rol de IAM para la ejecución de la máquina de estado

Defina la política de confianza asociada al rol de IAM para la ejecución de la máquina de estado.

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": "*"
                            }
                        ]
                    }
                }
            ]
        }
    },
```

Para crear una máquina de estado Lambda

Defina la máquina de estado Lambda.

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"
                    ]
                }
            }
        }
```

Paso 2: Utilice la AWS CloudFormation plantilla para crear una máquina de estado Lambda

Una vez que comprenda los componentes de la AWS CloudFormation plantilla, podrá unirlos y utilizarla para crear una AWS CloudFormation pila.

Para crear la máquina de estado Lambda

1. Copie los datos del siguiente ejemplo en un archivo denominado `MyStateMachine.yaml` para el ejemplo de YAML o `MyStateMachine.json` para el de 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

```
{
```

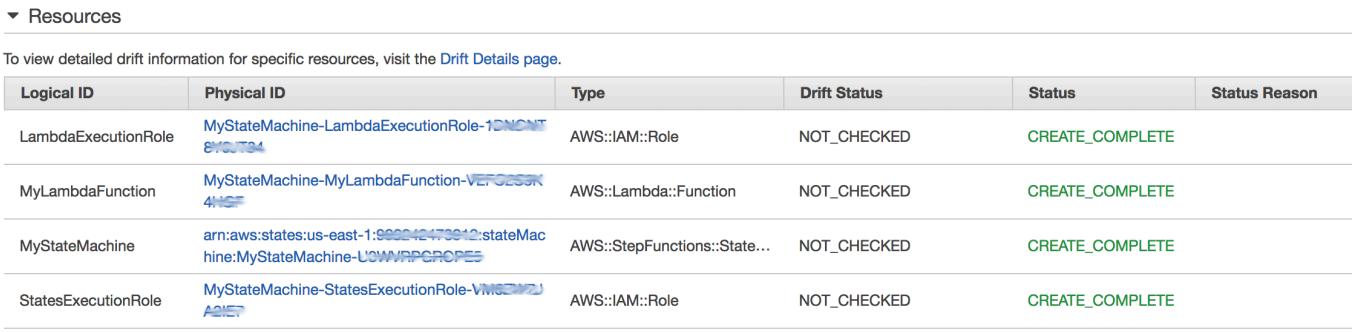
```
   "AWSTemplateFormatVersion": "2010-09-09",
   "Description": "An example template with an IAM role for a Lambda state
machine.",
   "Resources": {
       "LambdaExecutionRole": {
           "Type": "AWS::IAM::Role",
           "Properties": {
               "AssumeRolePolicyDocument": {
                   "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. Abra la consola de AWS CloudFormation y elija Crear pila.
3. En la página Seleccionar plantilla, elija Cargar una plantilla en Amazon S3. Elija su archivo `MyStateMachine` y después elija Siguiente.
4. En la página Especificar detalles, en Nombre de la pila, escriba `MyStateMachine` y, a continuación, elija Siguiente.
5. En la página Opciones, seleccione Siguiente.
6. En la página Revisar, seleccione Confirmo que AWS CloudFormation puede crear recursos de IAM. y, a continuación, elija Crear.

   AWS CloudFormation comienza a crear la `MyStateMachine` pila y muestra el estado CREATE_IN_PROGRESS. Cuando el proceso se haya completado, AWS CloudFormation mostrará el estado CREATE_COMPLETE.
7. (Opcional) Para mostrar los recursos de la pila, seleccione la pila y elija la pestaña Recursos.



▼ 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	

Paso 3: Iniciar la ejecución de una máquina de estado

Después de crear la máquina de estados Lambda, puede iniciar una ejecución.

Para iniciar la ejecución de la máquina de estado

1. Abra la consola Step Functions y elija el nombre de la máquina de estados que creó con ella AWS CloudFormation.
2. En la página ***MyStateMachine-ABCDEFGHIJ1K***, elija Nueva ejecución.

   Aparece la página Nueva ejecución.
3. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.
4. Seleccione Iniciar ejecución.

   Se inicia una nueva ejecución de la máquina de estado y aparece una nueva página que muestra la ejecución en funcionamiento.
5. (Opcional) En la sección Detalles de la ejecución, examine el Estado de ejecución y las marcas de tiempo Iniciada y Finalizada.

6. Para ver los resultados de la ejecución, elija Salida.

Creación de una máquina de estado Lambda para Step Functions mediante AWS CDK

En este tutorial se muestra cómo crear una máquina de estado de AWS Step Functions que contenga una función AWS Lambda mediante el AWS Cloud Development Kit (AWS CDK). AWS CDKEs un marco de infraestructura como código (IAC) que le permite definir la AWS infraestructura mediante un lenguaje de programación completo. Puede escribir una aplicación en uno de los lenguajes compatibles con CDK que contenga una o más pilas. A continuación, puede sintetizarlo en una AWS CloudFormation plantilla e implementarlo en su cuenta. AWS Usaremos este método para definir una máquina de Step Functions estados que contenga una Lambda función y, a continuación, utilizaremos el AWS Management Console para ejecutar la máquina de estados.

Antes de comenzar este tutorial, debe configurar su entorno de desarrollo de AWS CDK tal y como se describe en Getting Started With the AWS CDK - Prerequisites en la AWS Cloud Development Kit (AWS CDK) Developer Guide. A continuación, instale el AWS CDK con el siguiente comando en la AWS CLI:

```
npm install -g aws-cdk
```

Este tutorial produce el mismo resultado que the section called “Creación de una máquina de estados Lambda mediante AWS CloudFormation”. Sin embargo, en este tutorial, no se necesita que AWS CDK cree ningún rol de IAM; el AWS CDK lo hace por usted. La versión de AWS CDK también incluye un paso Succeed para ilustrar cómo añadir pasos adicionales a su máquina de estado.

> **Tip**
>
> Para implementar un ejemplo de aplicación sin servidor que inicie un Step Functions flujo de trabajo utilizando AWS CDK with TypeScript Cuenta de AWS, consulte el Módulo 10: Implementar con AWS CDK de The AWS Step Functions Workshop.

Temas

- Paso 1: Configurar el proyecto de AWS CDK
- Paso 2: Usar AWS CDK para crear una máquina de estado

Paso 1: Configurar el proyecto de AWS CDK

1. En su directorio principal, o en otro directorio si lo prefiere, ejecute el siguiente comando para crear un directorio para su nueva aplicación de AWS CDK.

   > **Important**
   >
   > Asegúrese de asignarle al directorio el nombre `step`. La plantilla de la aplicación de AWS CDK utiliza el nombre del directorio para generar nombres para los archivos y las clases fuente. Si utiliza otro nombre, la aplicación no coincidirá con este tutorial.

   TypeScript

   ```
   mkdir step && cd step
   ```

   JavaScript

   ```
   mkdir step && cd step
   ```

   Python

   ```
   mkdir step && cd step
   ```

   Java

   ```
   mkdir step && cd step
   ```

   C#

   Asegúrese de haber instalado la versión 6.0 o superior de .NET. Para obtener información, consulte Versiones compatibles.

```
mkdir step && cd step
```

2. Inicialice la aplicación mediante el comando cdk init. Especifique la plantilla ("aplicación") y el lenguaje de programación que desee, tal y como se muestra en los siguientes ejemplos.

   TypeScript

   ```
   cdk init --language typescript
   ```

   JavaScript

   ```
   cdk init --language javascript
   ```

   Python

   ```
   cdk init --language python
   ```

   Después de inicializar el proyecto, active el entorno virtual del proyecto e instale las dependencias de referencia del AWS CDK.

   ```
   source .venv/bin/activate
   python -m pip install -r requirements.txt
   ```

   Java

   ```
   cdk init --language java
   ```

   C#

   ```
   cdk init --language csharp
   ```

Paso 2: Usar AWS CDK para crear una máquina de estado

En primer lugar, presentaremos las piezas individuales de código que definen la función Lambda y la máquina de estado Step Functions. Luego, te explicaremos cómo unirlos en su aplicación de AWS CDK. Por último, verá cómo sintetizar e implementar estos recursos.

Para crear una función Lambda

El siguiente código de AWS CDK define la función Lambda y proporciona su código fuente en línea.

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)
});
```

En este breve ejemplo de código, puede ver lo siguiente:

- El nombre lógico de la función, `MyLambdaFunction`.
- El código fuente de la función, incrustado como una cadena en el código fuente de la aplicación de AWS CDK.
- Otros atributos de la función, como el tiempo de ejecución que se va a utilizar (Node 18.x), el punto de entrada de la función y el tiempo de espera.

Para crear una máquina de estado

Nuestra máquina de estado tiene dos estados: una tarea de función de Lambda, una tarea y un estado Succeed. La función requiere que creemos una the section called “Tarea” de Step Functions que invoque nuestra función. Este estado de tarea se usa como primer paso en la máquina de estado. El estado de éxito se agrega a la máquina de estado mediante el método `next()` del

estado de la tarea. El código siguiente invoca primero a la función nombrada `MyLambdaTask` y, a continuación, utiliza el método `next()` para definir un estado de éxito denominado `GreetedWorld`.

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")))
});
```

Para compilar e implementar la aplicación de AWS CDK

En el proyecto de AWS CDK recién creado, edite el archivo que contiene la definición de pila para que tenga el aspecto de uno de los siguientes códigos de ejemplo. Reconocerá las definiciones de la función de Lambda y la máquina de estado de Step Functions de las secciones anteriores.

1. Actualice la pila como se muestra en los siguientes ejemplos.

   TypeScript

   Actualice `lib/step-stack.ts` con el siguiente código.

```
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

Actualice `lib/step-stack.js` con el siguiente código.

```
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

Actualice `step/step_stack.py` con el siguiente código.

```
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

Actualice `src/main/java/com.myorg/StepStack.java` con el siguiente código.

```
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#

Actualice `scr/Step/StepStack.cs` con el siguiente código.

```
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. Guarde el archivo de código fuente y a continuación ejecute el comando `cdk synth` en el directorio principal de la aplicación.

   El AWS CDK ejecuta la aplicación y sintetiza una plantilla de AWS CloudFormation y, a continuación, AWS CDK la muestra.

   > **Note**
   >
   > Si solías TypeScript crear tu AWS CDK proyecto, al ejecutar el `cdk synth` comando se puede producir el siguiente error.
   >
   > ```
   > TSError: # Unable to compile TypeScript:
   > ```

```
bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
```

Vuelva a formalizar el archivo `bin/step.ts` como se muestra en el ejemplo siguiente para resolver este error.

```
#!/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. Para implementar la función de Lambda y la máquina de estado de Step Functions en su cuenta de AWS , procese `cdk deploy`. Se le pedirá que apruebe las políticas de IAM que AWS CDK haya generado.

Paso 3: Iniciar la ejecución de una máquina de estado

Después de crear la máquina de estado, puede iniciar su ejecución.

Para iniciar la ejecución de la máquina de estado

1. Abra la consola de Step Functions y elija el nombre de la máquina de estado que creó mediante AWS CDK.
2. En la página de la máquina de estado, seleccione Iniciar ejecución.

   Aparece el cuadro de diálogo Iniciar ejecución.
3. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que

puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

4. Seleccione Iniciar ejecución.

   Se inicia la ejecución de su máquina de estado y aparece una nueva página que muestra la ejecución en funcionamiento.

5. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Paso 4: Eliminación

Una vez que haya probado la máquina de estado, le recomendamos que la elimine junto con la función de Lambda relacionada para liberar recursos en su Cuenta de AWS. Ejecute el comando `cdk destroy` en el directorio principal de la aplicación para eliminar la máquina de estado.

Siguientes pasos

Para obtener más información sobre el desarrollo mediante el uso de AWS infraestructurasAWS CDK, consulte la Guía para AWS CDK desarrolladores.

Para obtener información acerca de cómo escribir aplicaciones AWS CDK en el lenguaje que elija, consulte:

TypeScript

- Trabajando AWS CDK con TypeScript

JavaScript

- Trabajando con AWS CDK in JavaScript

Python

Trabajar con AWS CDK en Python

Java

Trabajar con AWS CDK en Java

C#

Trabajar con AWS CDK en C#

Para obtener más información sobre los módulos de AWS Construct Library que se utilizan en este tutorial, consulte las siguientes descripciones generales de las referencias de AWS CDK API:

- aws-lambda
- aws-stepfunctions
- aws-stepfunctions-tasks

Creación de una API REST de API Gateway con Synchronous Express State Machine mediante AWS CDK

En este tutorial, se muestra cómo crear una API de REST de API Gateway con una máquina de estado rápida sincrónica como integración de backend mediante AWS Cloud Development Kit (AWS CDK). En este tutorial, se utilizará el constructo `StepFunctionsRestApi` para conectar la máquina de estado a la API Gateway. El constructo `StepFunctionsRestApi` configurará una asignación de entrada/salida predeterminado y la API de REST de API Gateway, con los permisos necesarios y un método HTTP “ANY”. AWS CDK Se trata de un marco de infraestructura como código (IAC) que le permite definir la AWS infraestructura mediante un lenguaje de programación completo. Escribes una aplicación en uno de los lenguajes compatibles con el CDK, que contiene una o más pilas, luego la sintetizas en una AWS CloudFormation plantilla y la despliegas en tu cuenta. AWS La usaremos para definir una API REST de API Gateway, que está integrada con Synchronous Express State Machine como backend, y luego la usaremos AWS Management Console para iniciar la ejecución.

Antes de iniciar este tutorial, configure su entorno de AWS CDK desarrollo tal y como se describe en Cómo empezar con los requisitos previos y, a continuación, instálelo de la siguiente manera AWS CDK : AWS CDK

```
npm install -g aws-cdk
```

Temas

- Paso 1: Configurar el proyecto de AWS CDK
- Paso 2: Úselo AWS CDK para crear una API REST de API Gateway con la integración de backend de Synchronous Express State Machine
- Paso 3: Comprobar la API Gateway
- Paso 4: Eliminación

Paso 1: Configurar el proyecto de AWS CDK

Primero, crea un directorio para tu nueva AWS CDK aplicación e inicializa el proyecto.

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
```

Una vez inicializado el proyecto, activa el entorno virtual del proyecto e instala las dependencias básicas AWS CDK del proyecto.

```
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**
>
> Asegúrese de asignarle al directorio el nombre `stepfunctions-rest-api`. La plantilla de la AWS CDK aplicación utiliza el nombre del directorio para generar nombres para las clases y los archivos fuente. Si utiliza otro nombre, la aplicación no coincidirá con este tutorial.

Ahora instale los módulos de la biblioteca de construcción para AWS Step Functions 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

Edite el archivo `pom.xml` del proyecto para agregar las siguientes dependencias dentro del contenedor `<dependencies>` existente.

```
        <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 instala automáticamente estas dependencias la próxima vez que cree su aplicación. Para la creación, ejecute `mvn compile` o utilice el comando Build (Crear) del IDE de Java.

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

También puede instalar los paquetes indicados mediante la NuGet GUI de Visual Studio, disponible en Tools > NuGet Package Manager > Manage NuGet Packages for Solution.

Una vez que haya instalado los módulos, podrá utilizarlos en su AWS CDK aplicación importando los siguientes paquetes.

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

Añada lo siguiente a `import` dentro de `stepfunctions-rest-api.go`.

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

Paso 2: Úselo AWS CDK para crear una API REST de API Gateway con la integración de backend de Synchronous Express State Machine

En primer lugar, presentaremos los fragmentos de código individuales que definen la máquina de estado express sincrónica y la API REST de API Gateway y, a continuación, explicaremos cómo unirlos en su AWS CDK aplicación. A continuación, verá cómo sintetizar e implementar estos recursos.

> **Note**
>
> La máquina de estado que mostraremos aquí será una máquina de estado simple con un estado `Pass`.

Para crear una máquina de estado rápida

Este es el AWS CDK código que define una máquina de estados simple con un `Pass` estado.

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,
})
```

En este breve fragmento de código, puede ver lo siguiente:

- La definición de máquina denominada `PassState`, que es un estado `Pass`.
- El nombre lógico de la máquina de estado, `MyStateMachine`.
- La definición de la máquina se utiliza como definición de la Máquina de estado.
- El tipo de máquina de estado se establece como `EXPRESS` porque `StepFunctionsRestApi` solo permitirá una máquina de estado rápida síncrona.

Para crear una API de REST de API Gateway utilizando constructo **`StepFunctionsRestApi`**

Usaremos el constructo `StepFunctionsRestApi` para crear la API de REST de API Gateway con los permisos necesarios y la asignación de entrada/salida predeterminada.

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,
})
```

Para compilar e implementar la aplicación de AWS CDK

En el AWS CDK proyecto que has creado, edita el archivo que contiene la definición de la pila para que se parezca al código de abajo. Reconocerá las definiciones de la máquina de estado de Step Functions y la API Gateway de más arriba.

TypeScript

Actualice `lib/stepfunctions-rest-api-stack.ts` para que diga lo siguiente.

```
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

Actualice `lib/stepfunctions-rest-api-stack.js` para que diga lo siguiente.

```
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

Actualice `stepfunctions_rest_api/stepfunctions_rest_api_stack.py` para que diga lo siguiente.

```
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

Actualice `src/main/java/com.myorg/StepfunctionsRestApiStack.java` para que diga lo siguiente.

```
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#

Actualice `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs` para que diga lo siguiente.

```
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

Actualice `stepfunctions-rest-api.go` para que diga lo siguiente.

```
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")),
```

```
    // }
}
```

Guarde el archivo fuente y, a continuación, ejecute `cdk synth` en el directorio principal de la aplicación. AWS CDK Ejecuta la aplicación, sintetiza una AWS CloudFormation plantilla a partir de ella y, a continuación, muestra la plantilla.

Para implementar realmente Amazon API Gateway y la máquina de AWS Step Functions estados en su cuenta de AWS, emita`cdk deploy`. Se le pedirá que apruebe las políticas de IAM que AWS CDK haya generado. Las políticas que se van a crear tendrán un aspecto similar al siguiente:

```
IAM Statement Changes
┌───┬──────────────────────────────────────┬────────┬──────────────────────────────────┬──────────────────────────────────┬───────────┐
│   │ Resource                             │ Effect │ Action                           │ Principal                        │ Condition │
├───┼──────────────────────────────────────┼────────┼──────────────────────────────────┼──────────────────────────────────┼───────────┤
│ + │ ${SfnDemoCdkStack--StateMachine-apiRol │ Allow  │ sts:AssumeRole                   │ Service:apigateway.amazonaws.com │           │
│   │ e.Arn}                               │        │                                  │                                  │           │
├───┼──────────────────────────────────────┼────────┼──────────────────────────────────┼──────────────────────────────────┼───────────┤
│ + │ ${StateMachine}                      │ Allow  │ states:StartSyncExecution        │ AWS:${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)?
```

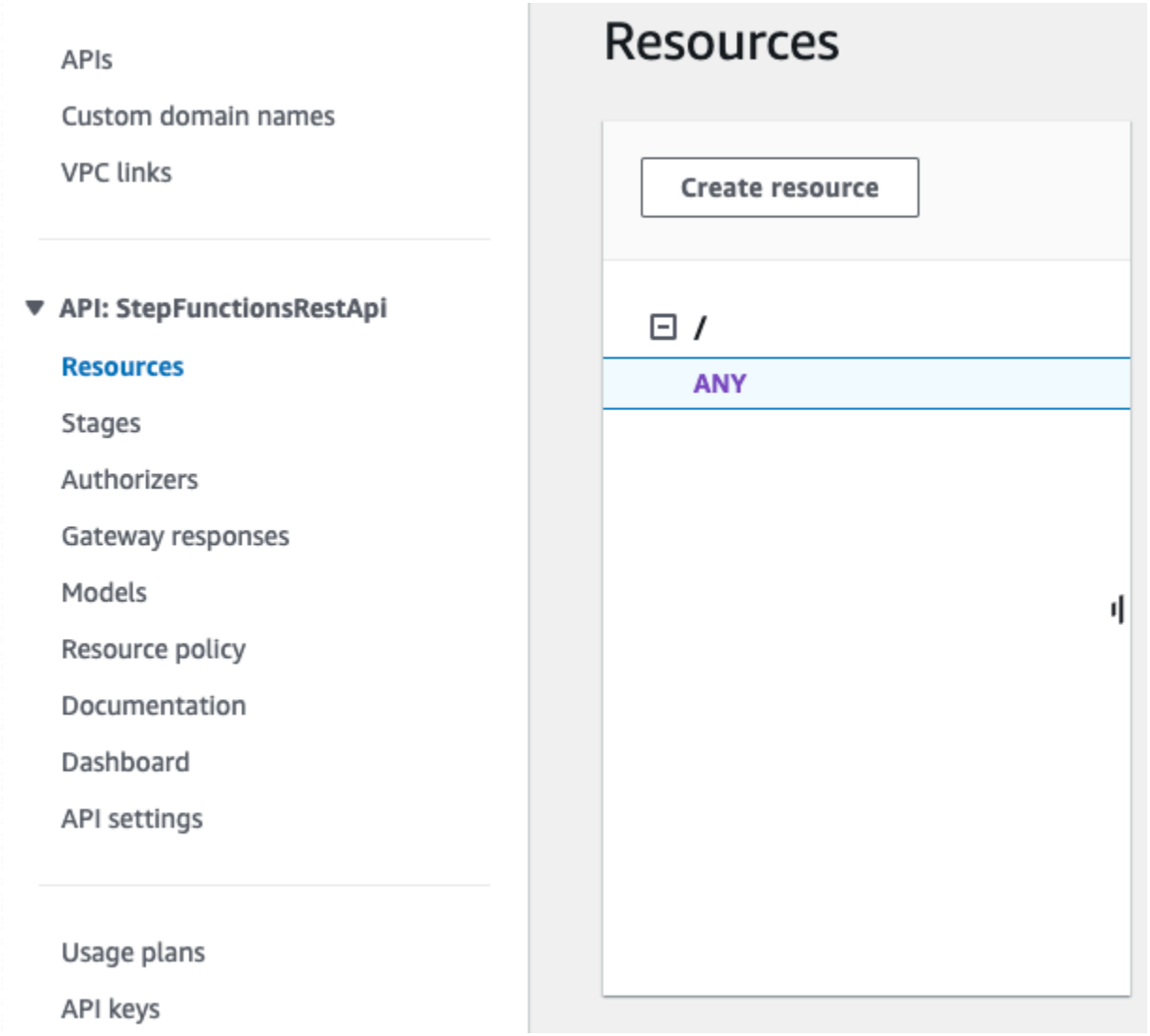
Paso 3: Comprobar la API Gateway

Después de crear la API de REST de API Gateway con máquina de estado rápida sincrónica como integración de backend, puede probar la API Gateway.

Para probar la API Gateway implementada mediante la consola de API Gateway

1. Abra la consola de Amazon API Gateway e inicie sesión.
2. Elija el nombre de su API de REST denominada `StepFunctionsRestApi`.

3. En el panel Recursos, elija el método ANY.



4. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
5. En Method (Método), elija POST.
6. En Cuerpo de la solicitud, copie los siguientes parámetros de solicitud.

```
{
    "key": "Hello"
}
```

7. Seleccione Test (Probar). Se mostrará la siguiente información:

   - Request (Solicitud) es la ruta del recurso llamada para el método.
   - Status (Estado) es el código de estado HTTP de la respuesta.

- Latency (Latencia) es el tiempo entre la recepción de la solicitud del intermediario y la respuesta devuelta.
- Cuerpo de respuesta es el cuerpo de la respuesta HTTP.
- Encabezados de respuesta son los encabezados de respuesta HTTP.
- El registro muestra las entradas simuladas de Amazon CloudWatch Logs que se habrían escrito si se hubiera llamado a este método fuera de la consola de API Gateway.

> **Note**
>
> Aunque las entradas de CloudWatch Logs son simuladas, los resultados de la llamada al método son reales.

La salida de Cuerpo de respuesta debe tener un aspecto similar al siguiente:

```
"Hello"
```

> **Tip**
>
> Pruebe la API Gateway con métodos diferentes y una entrada no válida para ver el resultado del error. Es posible que desee cambiar la máquina de estado para buscar una clave en particular y, durante las pruebas, proporcionar la clave incorrecta para que no se ejecute correctamente la máquina de estado y generar un mensaje de error en la salida de Cuerpo de respuesta.

Para probar la API implementada mediante cURL

1. Abra una ventana de terminal.
2. Copie el siguiente comando cURL y péguelo en la ventana de terminal, sustituyendo `<api-id>` por el ID de la API de la API y `<region>` por la región en la que se implementa la API.

```
curl -X POST\
 'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
 -d '{"key":"Hello"}' \
 -H 'Content-Type: application/json'
```

El resultado del cuerpo de la respuesta debe tener un aspecto similar al siguiente:

```
"Hello"
```

> **Tip**
>
> Pruebe la API Gateway con métodos diferentes y una entrada no válida para ver el resultado del error. Es posible que desee cambiar la máquina de estado para buscar una clave en particular y, durante las pruebas, proporcionar la clave incorrecta para que no se ejecute correctamente la máquina de estado y generar un mensaje de error en el resultado del cuerpo de la respuesta.

Paso 4: Eliminación

Cuando termine de probar su API Gateway, podrá desmantelar tanto la máquina de estado como la API Gateway con el AWS CDK. Ejecute `cdk destroy` en el directorio principal de la aplicación.

AWS Step Functions SDK de ciencia de datos para Python

El SDK AWS Step Functions de ciencia de datos es una biblioteca de código abierto para científicos de datos. Con este SDK, puede crear flujos de trabajo que procesen y publiquen modelos de aprendizaje automático mediante SageMaker Step Functions. También puede crear flujos de trabajo de aprendizaje automático de varios pasos en Python que organicen la AWS infraestructura a escala, sin tener que aprovisionar e integrar los AWS servicios por separado.

El SDK de AWS Step Functions Data Science proporciona una API de Python que puede crear e invocar flujos de trabajo de Step Functions. Puede administrar y ejecutar estos flujos de trabajo directamente en Python, así como en cuadernos de Jupyter.

Además de crear flujos de trabajo listos para la producción directamente en Python, el SDK de AWS Step Functions Data Science le permite copiar ese flujo de trabajo, experimentar con nuevas opciones y, después, poner el flujo de trabajo refinado en producción.

Para obtener más información sobre el SDK de ciencia de AWS Step Functions datos, consulte lo siguiente:

- Proyecto en Github

- Documentación de SDK
- Los siguientes ejemplos de cuadernos, que están disponibles en las instancias de Jupyter Notebook en la SageMaker consola y en el proyecto relacionado: GitHub
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

Implementar máquinas de estado con Terraform

Terraform de HashiCorp es un marco para crear aplicaciones que utilizan la infraestructura como código (IaC). Con Terraform, puede crear máquinas de estado y utilizar características, como obtener una vista previa de las implementaciones de infraestructura y crear plantillas reutilizables. Las plantillas de Terraform le ayudan a mantener y reutilizar el código al dividirlo en partes más pequeñas.

Si está familiarizado con Terraform puede seguir el ciclo de vida de desarrollo descrito en este tema como modelo para crear e implementar máquinas de estado en Terraform. Si no está familiarizado con Terraform le recomendamos que antes de comenzar realice el taller Introducción a Terraform en AWS para familiarizarse con Terraform.

> **Tip**
>
> Para implementar un ejemplo de máquina de estado creada con Terraform en Cuenta de AWS, consulte el módulo Administrar máquinas de estado con infraestructura como código de The AWS Step Functions Workshop.

En este tema

Requisitos previos

Antes de comenzar debe cumplir los siguientes requisitos previos:

- Instale Terraform en su máquina. Para obtener información sobre la instalación de Terraform, consulte Instalar Terraform.
- Instale Step Functions Local en su máquina. Se recomienda instalar la imagen de Docker de Step Functions Local para utilizar Step Functions Local. Para obtener más información, consulte Probar máquinas de estado de forma local.
- Instalación de la AWS SAM CLI. Para obtener información sobre instalación, consulte Instalación del AWS SAM CLI en la Guía para desarrolladores de AWS Serverless Application Model.
- Instale el AWS Toolkit for Visual Studio Code para ver el diagrama de flujo de trabajo de sus máquinas de estado. Para obtener información sobre la instalación, consulte Instalación del AWS Toolkit for Visual Studio Code en la Guía del usuario de AWS Toolkit for Visual Studio Code.

Ciclo de vida del desarrollo de máquinas de estado con Terraform

El siguiente procedimiento explica cómo puede utilizar un prototipo de máquina de estado que se crea con Workflow Studio en la consola de Step Functions como punto de partida para desarrollo local con Terraform y el AWS Toolkit for Visual Studio Code.

Para ver el ejemplo completo que analiza el desarrollo de máquinas de estado con Terraform y presenta las mejores prácticas en detalle, consulte Best practices for writing Step Functions Terraform projects.

Para iniciar el ciclo de vida de desarrollo de una máquina de estado con Terraform

1. Inicie un nuevo proyecto de Terraform con el siguiente comando.

   ```
   terraform init
   ```

2. Abra la consola de Step Functions para crear un prototipo para la máquina de estado.
3. En Workflow Studio, haga lo siguiente:

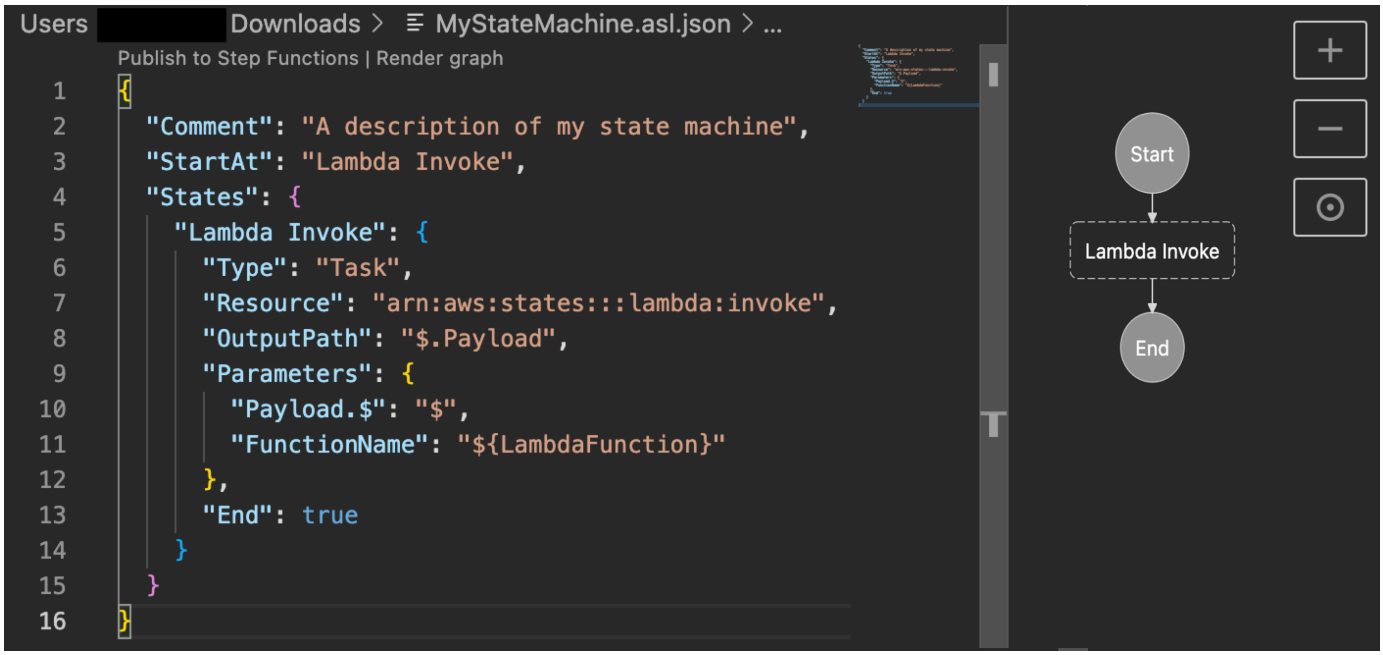
   a. Cree el prototipo del flujo de trabajo.

   b. Exporte la definición de Amazon States Language (ASL) del flujo de trabajo. Para ello, seleccione la lista desplegable Importar/Exportar y, a continuación, seleccione Exportar definición de JSON.
4. Guarde la definición de ASL exportada en el directorio del proyecto.

La definición de ASL exportada se transfiere como parámetro de entrada al recurso de Terraform de `aws_sfn_state_machine` que utiliza la función `templatefile`. Esta función se utiliza en el campo de definición que transmite la definición de ASL exportada y cualquier sustitución de variables.

> **Tip**
>
> Como el archivo de definición de ASL puede contener bloques de texto largos, le recomendamos que evite el método EOF insertado. De este modo resulta más sencillo sustituir los parámetros en la definición de la máquina de estado.

5. (Opcional) Actualice la definición de ASL en su IDE y visualice los cambios mediante AWS Toolkit for Visual Studio Code.



Para evitar exportar continuamente la definición y refactorizarla en el proyecto, le recomendamos que realice las actualizaciones localmente en su IDE y que haga un seguimiento de estas actualizaciones con Git.

6. Pruebe el flujo de trabajo utilizando Step Functions Local.

> **Tip**
>
> También puede probar localmente las integraciones de servicios con las funciones de Lambda y las API de API Gateway en la máquina de estado mediante AWS SAM CLI Local.

7. Obtenga una vista previa de la máquina de estado y otros recursos de AWSantes de implementar la máquina de estado. Para ello, ejecute el siguiente comando.

```
terraform plan
```

8. Implemente la máquina de estado desde su entorno local o mediante canalizaciones de CI/CD utilizando el siguiente comando.

```
terraform apply
```

9. (Opcional) Limpie los recursos y elimine la máquina de estado con el siguiente comando.

```
terraform destroy
```

Políticas y roles de IAM para la máquina de estado

Utilice las políticas de integración de servicios de Terraform para añadir los permisos de IAM necesarios a la máquina de estado, por ejemplo, permiso para invocar funciones de Lambda. También puede definir roles y políticas explícitos y asociarlos a la máquina de estado.

El siguiente ejemplo de política de IAM concede acceso a la máquina de estado para invocar una función de Lambda llamada *myFunction*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
```

```
  ]
}
```

También recomendamos utilizar el origen de datos `aws_iam_policy_document` al definir las políticas de IAM para máquinas de estado en Terraform. Le ayudará a comprobar si su política está mal estructurada y a sustituir los recursos por variables.

El siguiente ejemplo de política de IAM utiliza el origen de datos `aws_iam_policy_document` y concede acceso a la máquina de estado para invocar una función de Lambda llamada *`myFunction`*.

```
data "aws_iam_policy_document" "state_machine_role_policy" {

  statement {
    effect = "Allow"

    actions = [
      "lambda:InvokeFunction"
    ]

    resources = ["${aws_lambda_function.[[myFunction]].arn}:*"]
  }

}
```

> **Tip**
>
> Para ver patrones arquitectónicos de AWS más avanzados implementados con Terraform, consulte los Ejemplos de Terraform en Serverless Land Workflows Collection.

Prueba y depuración

Step Functions proporciona diferentes formas de probar y depurar máquinas de estado. Por ejemplo, puede probar y depurar las máquinas de estado en la consola, utilizar la API TestState para probar un estado individual o utilizar Step Functions Local para probar las máquinas de estado de forma local.

Con la API TestState, usted facilita la definición de un solo estado y la ejecuta. Puede probar un solo estado sin crear una máquina de estado ni actualizar una máquina de estado existente.

Step Functions Local es una versión descargable de Step Functions que permite desarrollar y probar aplicaciones utilizando una versión de Step Functions que se ejecuta en su propio entorno de desarrollo. Con Step Functions Local, puede ejecutar sus máquinas de estado para probar sus flujos de datos de entrada y salida, integraciones con servicios compatibles y mucho más en su entorno de desarrollo local.

Temas

Uso TestState de la API para probar un estado

La TestStateAPI acepta la definición de un solo estado y la ejecuta. Puede probar un estado sin crear una máquina de estado ni actualizar una máquina de estado existente.

Con la TestState API, puede probar lo siguiente:

- El flujo de datos de procesamiento de entrada y salida de un estado.
- Una Servicio de AWS integración con otras Servicios de AWS solicitudes y respuestas
- La solicitud y respuesta de una tarea HTTP

Para probar un estado, también puede usar la consola de Step Functions, AWS Command Line Interface (AWS CLI) o el SDK.

La API `TestState` asume un rol de IAM que debe contener los permisos de IAM necesarios para los recursos a los que accede su estado. Para obtener información acerca de los permisos que podría necesitar un estado, consulte IAMpermisos para usar la TestState API.

Temas

Consideraciones sobre el uso de la TestState API

Con la TestStateAPI, solo puede probar un estado a la vez. Los estados que puede probar son los siguientes:

- Todos los Tipos de tareas, salvo Actividades
- Pass
- Wait
- Choice
- Succeed
- Fail

Cuando utilice la API `TestState`, tenga en cuenta las siguientes consideraciones.

- La TestState API no admite lo siguiente:
  - Estados de Estado de la tarea que utilizan los siguientes tipos de recursos:
    - Actividad
    - Patrones de integración de servicios de tipo `.sync` o `.waitForTaskToken`
  - Estado de Parallel
  - Estado de Map
- Una prueba puede ejecutarse durante un máximo de cinco minutos. Si una prueba supera esta duración, fallará y mostrará el error `States.Timeout`.

Uso de niveles de inspección en la TestState API

Para probar un estado mediante la TestStateAPI, debes proporcionar la definición de ese estado. A continuación, la prueba devuelve un resultado. Para cada estado, puede especificar la cantidad de detalles que desea ver en los resultados de la prueba. Estos detalles proporcionan información adicional acerca del estado que está probando. Por ejemplo, si ha utilizado filtros de procesamiento de datos de entrada y salida, como `InputPath` o `ResultPath` en un estado, puede ver los resultados del procesamiento de datos intermedio y final.

Step Functions proporciona los siguientes niveles para especificar los detalles que desea ver:

- INFO
- DEBUG
- TRACE

Todos estos niveles también devuelven los campos `status` y `nextState`. `status` indica el estado de la ejecución de estado. Por ejemplo, `SUCCEEDED`, `FAILED`, `RETRIABLE` y `CAUGHT_ERROR`. `nextState` indica el nombre del siguiente estado al que se realizará la transición. Si no ha definido un estado siguiente en la definición, este campo devuelve un valor vacío.

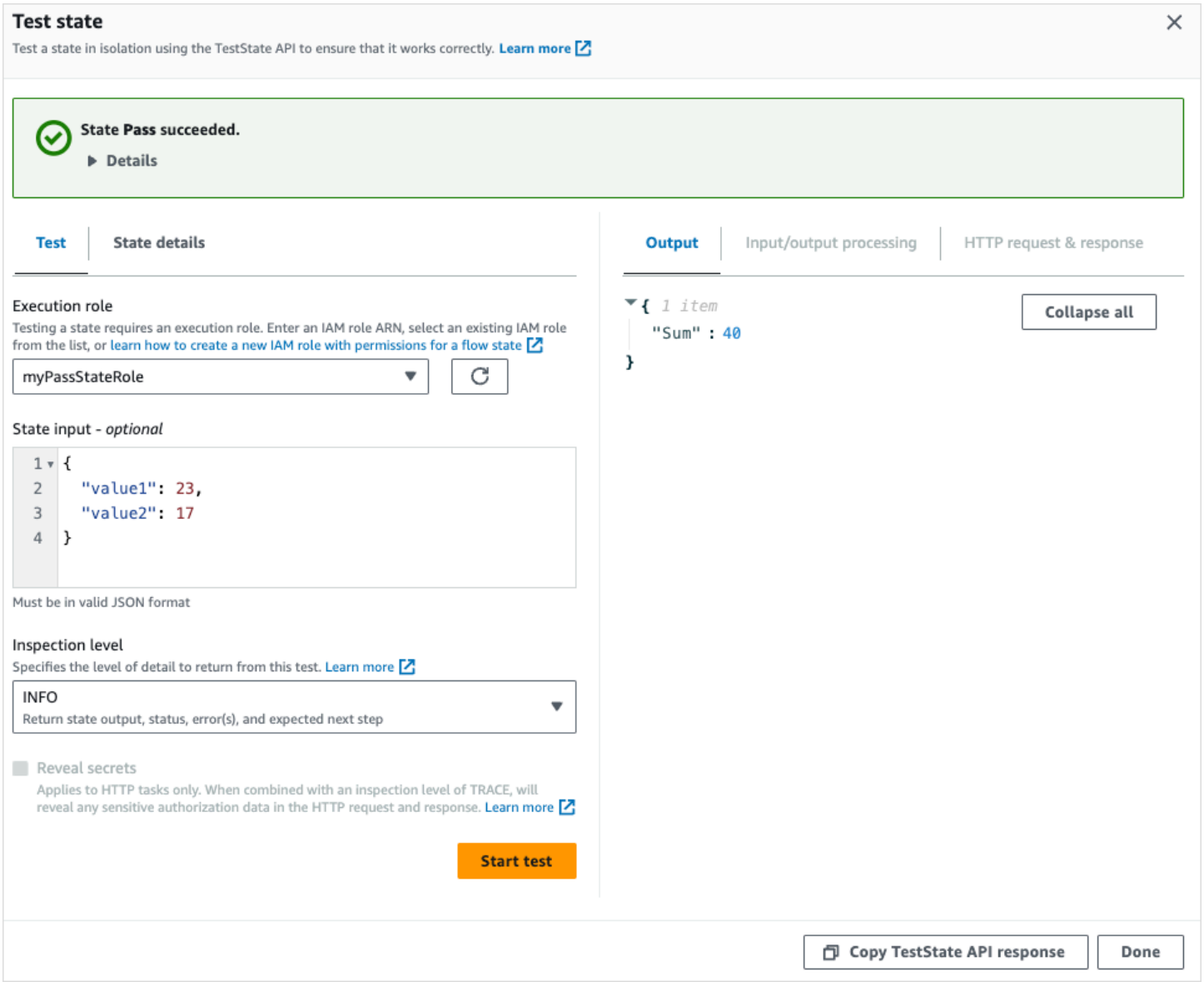
Para obtener información sobre cómo probar un estado mediante estos niveles de inspección en la consola de Step Functions y AWS CLI, consulte Probar un estado (consola) y Probar un estado mediante AWS CLI.

INFO inspectionLevel

Si la prueba se realiza correctamente, este nivel muestra la salida de estado. Si la prueba falla, este nivel muestra la salida de error. De forma predeterminada, Step Functions establece Nivel de inspección en INFO si no se especifica ningún nivel.

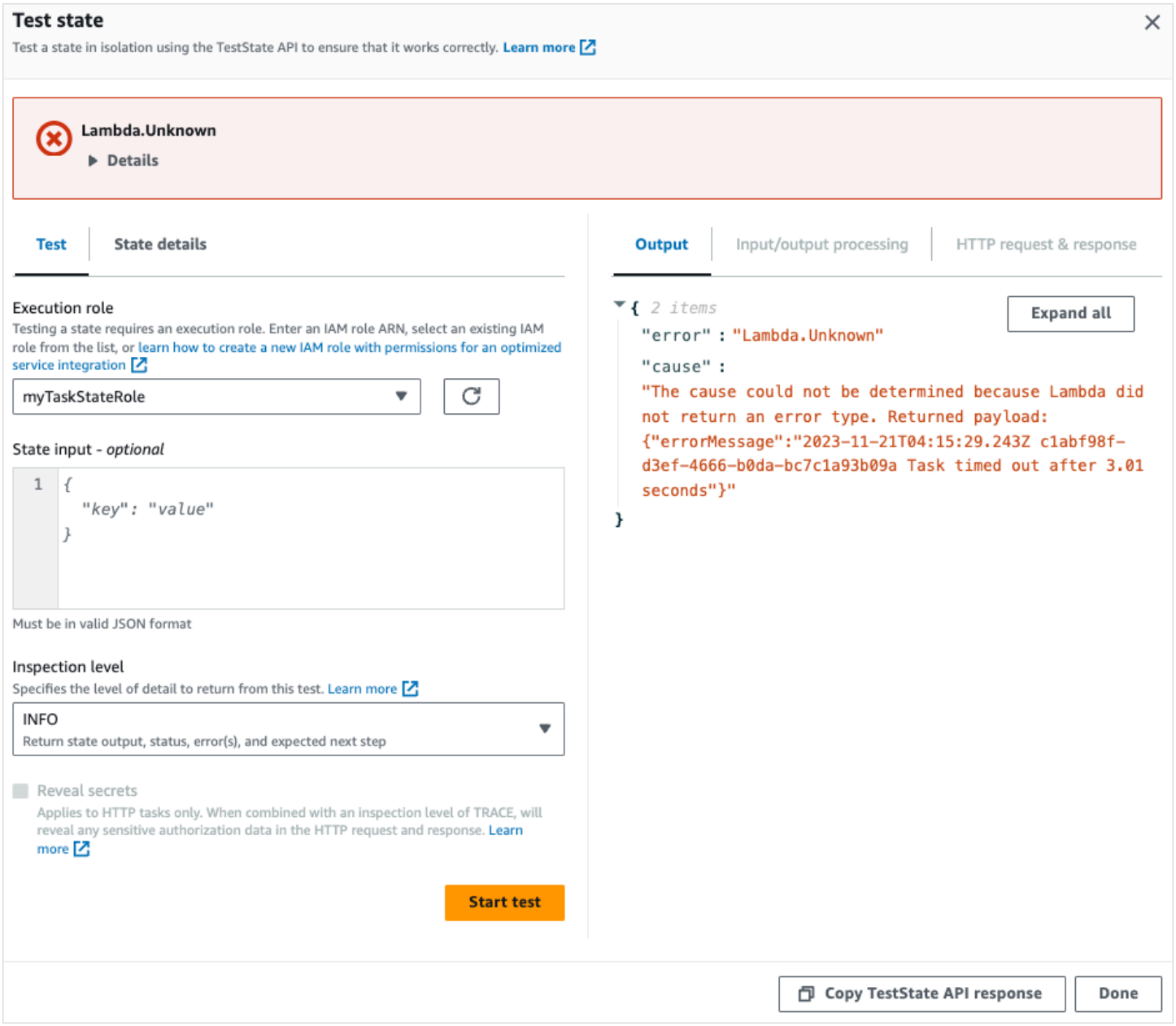
Ejemplo de prueba con nivel INFO que produce un resultado correcto

En la siguiente imagen se muestra una prueba para un estado Aprobado que se ha realizado correctamente. El Nivel de inspección de este estado se estable en INFO y el resultado del estado aparece en la pestaña Salida.



Ejemplo de prueba con nivel INFO con error

La siguiente imagen muestra una prueba que falló para un estado Tarea cuando el Nivel de inspección se establece en INFO. La pestaña Salida muestra la salida de error, que incluye el nombre del error y una explicación detallada de la causa del error.



DEBUG inspectionLevel

Si la prueba se realiza correctamente, este nivel muestra la salida de estado y el resultado del procesamiento de datos de entrada y salida.

Si la prueba falla, este nivel muestra la salida de error. Este nivel muestra los resultados intermedios del procesamiento de datos hasta el punto de error. Por ejemplo, supongamos que probó un estado Tarea que invoca una función de Lambda. Supongamos que había aplicado los filtros InputPath, Parámetros, ResultPath y OutputPath al estado Tarea. Supongamos que la invocación ha producido

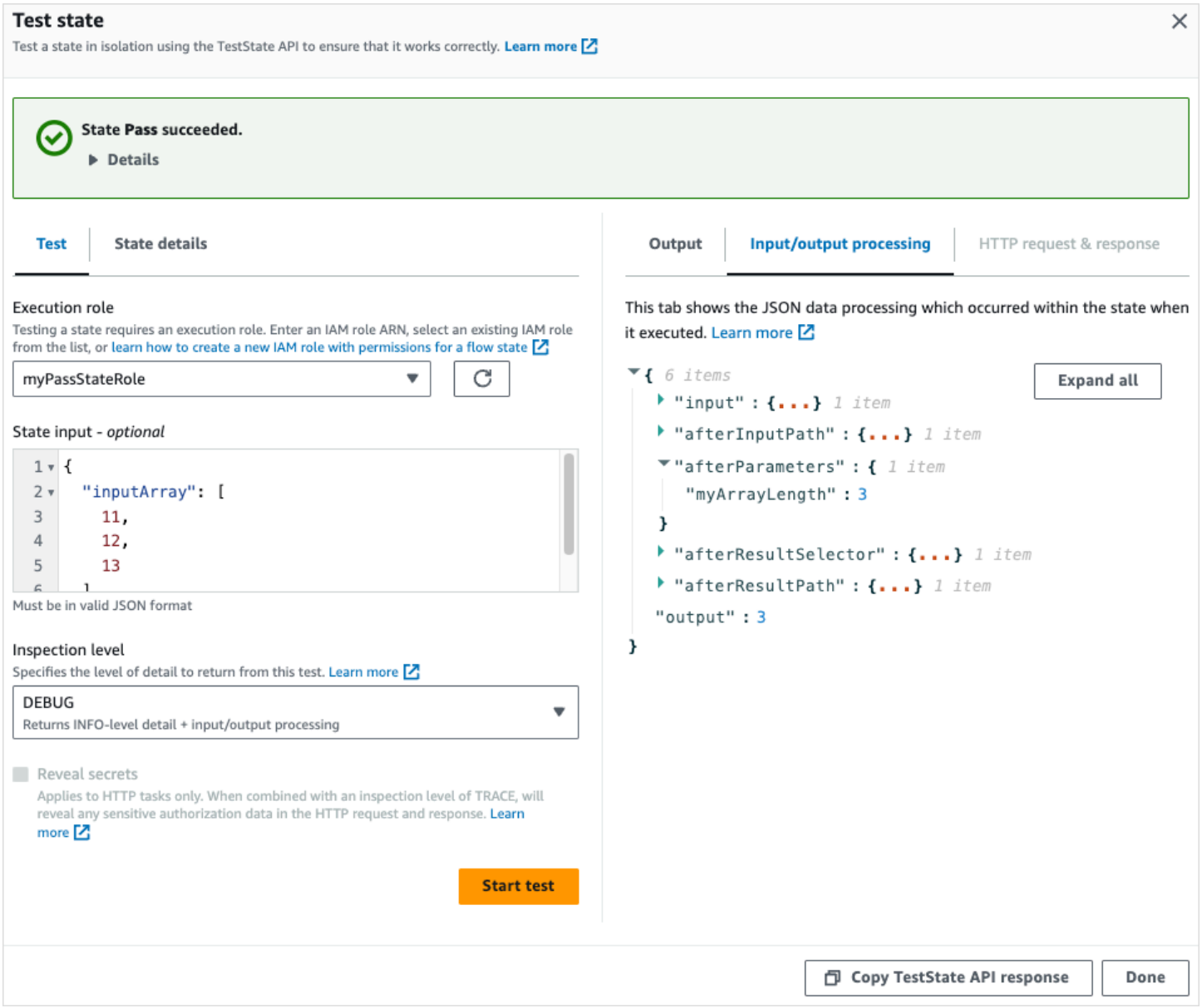
error. En este caso, el nivel `DEBUG` muestra los resultados del procesamiento de datos en función de la aplicación de los filtros en el siguiente orden:

- `input`: entrada de estado sin procesar
- `afterInputPath`: la entrada después de Step Functions aplica el filtro `InputPath`.
- `afterParameters`: la entrada efectiva después de Step Functions aplica el filtro `Parameters`.

La información de diagnóstico disponible en este nivel puede ayudar a solucionar problemas relacionados con un flujo de integración de servicios o de procesamiento de datos de entrada y salida que haya definido.

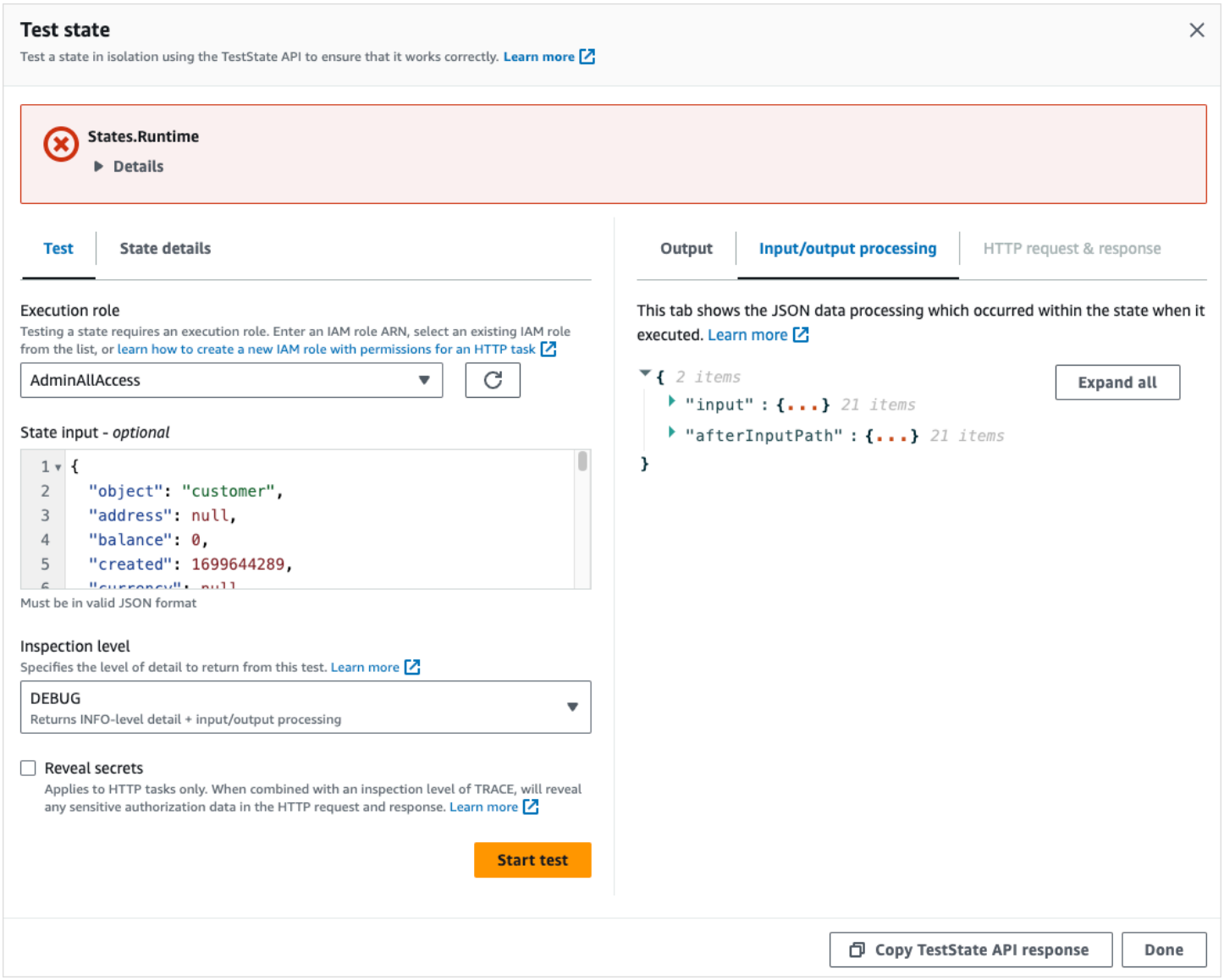
Ejemplo de prueba con nivel DEBUG que produce un resultado correcto

En la siguiente imagen se muestra una prueba para un estado Aprobado que se ha realizado correctamente. El Nivel de inspección de este estado se establece en DEBUG. La pestaña Procesamiento de entrada/salida de la imagen siguiente muestra el resultado de la aplicación de `Parameters` a la entrada proporcionada para este estado.



Ejemplo de prueba con nivel DEBUG con error

La siguiente imagen muestra una prueba que falló para un estado Tarea cuando el Nivel de inspección se establece en DEBUG. La pestaña Procesamiento de entrada/salida de la imagen siguiente muestra el resultado del procesamiento de los datos de entrada y salida del estado hasta el punto de error.



TRACE inspectionLevel

Step Functions proporciona el nivel TRACE para probar una Tarea HTTP. Este nivel devuelve información sobre la solicitud HTTP que Step Functions realiza y la respuesta que devuelve una API de terceros. La respuesta puede contener información, como los encabezados y el cuerpo de la solicitud. Además, puede ver la salida del estado y el resultado del procesamiento de datos de entrada y salida en este nivel.

Si la prueba falla, este nivel muestra la salida de error.

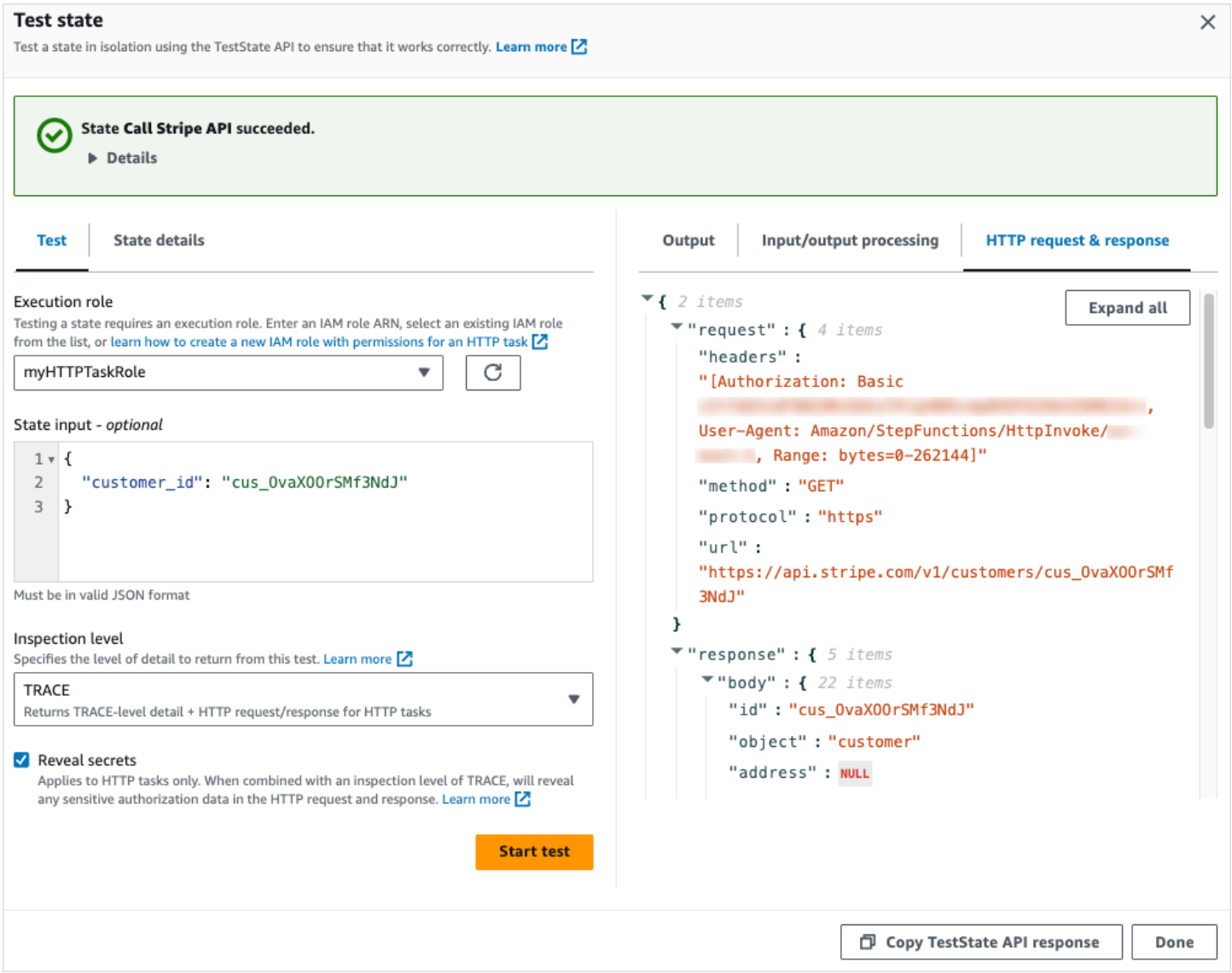
Este nivel solo es aplicable para una tarea HTTP. Step Functions presenta un error si utiliza este nivel para otros tipos de estado.

Al establecer el nivel de inspección en TRACE, también puede ver los secretos incluidos en la EventBridge conexión. Para ello, debe configurar el `revealSecrets` parámetro `true` en la TestStateAPI. Además, debes asegurarte de que el IAM usuario que llama a la TestState API tiene permiso para realizar la `states:RevealSecrets` acción. Para ver una política de IAM de ejemplo que establece el permiso `states:RevealSecrets`, consulte IAMpermisos para usar la TestState API. Sin este permiso, Step Functions produce un error de acceso denegado.

Si establece el parámetro `revealSecrets` en `false`, Step Functions omite todos los secretos de los datos de solicitud y respuesta HTTP.

Ejemplo de prueba con nivel TRACE que produce un resultado correcto

En la siguiente imagen se muestra una prueba para una tarea HTTP que se ha realizado correctamente. El Nivel de inspección de este estado se establece en TRACE. La pestaña Solicitud y respuesta HTTP de la siguiente imagen muestra el resultado de la llamada a la API de terceros.



IAMpermisos para usar la TestState API

El usuario de IAM que llama a la API `TestState` debe tener permisos para realizar las acciones `states:TestState` y `iam:PassRole`. Además, si establece el parámetro revealSecrets en `true`, debe asegurarse de que el usuario de IAM tiene permisos para realizar la acción `states:RevealSecrets`. Sin este permiso, Step Functions produce un error de acceso denegado.

También debe asegurarse de que su rol de ejecución contiene los permisos IAM necesarios para los recursos a los que accede su estado. Para obtener información acerca de los permisos que podría necesitar un estado, consulte Administración de roles de ejecución.

La siguiente política de IAM de ejemplo establece los permisos `states:TestState`, `iam:PassRole` y `states:RevealSecrets`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

Probar un estado (consola)

Puede probar un estado en la consola y comprobar la salida del estado o el flujo de procesamiento de datos de entrada y salida. En el caso de una Tarea HTTP, puede probar la solicitud y la respuesta HTTP sin procesar.

Para probar un estado

1. Abra la consola de Step Functions.
2. Seleccione Crear máquina de estado para empezar a crear una máquina de estado o elija una máquina de estado existente.
3. En el Modo Diseño de Workflow Studio, elija un estado que desea probar.
4. Elija Probar estado en el panel de Inspector de Workflow Studio.
5. En el cuadro de diálogo Probar estado, haga lo siguiente:

    a. En Rol de ejecución, elija un rol de ejecución para probar el estado. Asegúrese de que cuenta con los permisos de IAM necesarios para el estado que desea probar.

    b. (Opcional) Proporcione cualquier entrada JSON que necesite el estado seleccionado para la prueba.

    c. En Nivel de inspección, seleccione una de las siguientes opciones en función de los valores que desee ver:

    - INFO: muestra la salida del estado en la pestaña Salida si la prueba se realiza correctamente. Si la prueba produce error, INFO muestra la salida de error, que

incluye el nombre del error y una explicación detallada de la causa del error. De forma predeterminada, Step Functions establece Nivel de inspección en INFO si no se selecciona ningún nivel.

- DEBUG: muestra la salida de estado y el resultado del procesamiento de datos de entrada y salida si la prueba se realiza correctamente. Si la prueba produce error, DEBUG muestra la salida de error, que incluye el nombre del error y una explicación detallada de la causa del error.
- TRACE: muestra la solicitud y la respuesta HTTP sin procesar y es útil para verificar encabezados, parámetros de consulta y otros detalles específicos de la API. Esta opción solo está disponible para la Tarea HTTP.

  Si lo desea, puede seleccionar Revelar secretos. En combinación con TRACE, esta configuración permite ver los datos confidenciales que inserta la conexión de EventBridge, como claves de API. La identidad del usuario de IAM que utilice para acceder a la consola debe tener permiso para realizar la acción `states:RevealSecrets`. Sin este permiso, Step Functions produce un error de acceso denegado al iniciar la prueba. Para ver una política de IAM de ejemplo que establece el permiso `states:RevealSecrets`, consulte IAMpermisos para usar la TestState API.

d. Seleccione Iniciar prueba.

Probar un estado mediante AWS CLI

Puede probar un estado compatible mediante la TestStateAPI deAWS CLI. La API acepta la definición de un estado y la ejecuta.

Para cada estado, puede especificar la cantidad de detalles que desea ver en los resultados de la prueba. Estos detalles proporcionan información adicional sobre la ejecución del estado, incluido el resultado del procesamiento de datos de entrada y salida y la información de solicitud y respuesta HTTP. Los siguientes ejemplos muestran los diferentes niveles de inspección que puede especificar para la TestState API. Recuerde reemplazar el texto en *`cursiva`* por la información específica del recurso.

Esta sección contiene los siguientes ejemplos que describen cómo puede utilizar los diferentes niveles de inspección que Step Functions proporciona en AWS CLI:

- Uso de INFO inspectionLevel
- Uso de DEBUG inspectionLevel

- Uso de TRACE inspectionLevel
- Uso de la utilidad jq AWS CLI para filtrar e imprimir la respuesta HTTP que devuelve la TestState API

Ejemplo 1: Uso del InspectionLevel INFO para probar un estado Elección

Para probar un estado mediante el `INFO` InspectionLevel del AWS CLI, ejecuta el `test-state` comando como se muestra en el siguiente ejemplo.

```
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}'
```

En este ejemplo se utiliza un estado Elección para determinar la ruta de ejecución del estado en función de la entrada numérica que proporcione. De forma predeterminada, Step Functions establece el `inspectionLevel` en `INFO` si no se establece un nivel.

Step Functions devuelve la siguiente salida.

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

Ejemplo 2: Uso del InspectionLevel DEBUG para depurar el procesamiento de datos de entrada y salida en un estado Aprobado

Para probar un estado mediante el `DEBUG` InspectionLevel del AWS CLI, ejecute el `test-state` comando como se muestra en el siguiente ejemplo.

```
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
```

En este ejemplo se usa un estado Pass para mostrar cómo Step Functions filtra y manipula los datos JSON de entrada mediante los filtros de procesamiento de datos de entrada y salida. En este ejemplo se utilizan los siguientes filtros: `InputPath`, `Parámetros`, `ResultPath` y `OutputPath`.

Step Functions devuelve la siguiente salida.

```
{
    "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"
}
```

Ejemplo 3: Uso del InspectionLevel TRACE y revealSecrets para inspeccionar la solicitud HTTP enviada a una API de terceros

Para probar una tarea HTTP mediante el TRACE InspectionLevel junto con el parámetro RevealSecrets del AWS CLI, ejecute el `test-state` comando como se muestra en el siguiente ejemplo.

```
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
```

En este ejemplo se comprueba si la tarea HTTP llama a la API de terceros especificada, `https://httpbin.org/`. También se muestran los datos de solicitud y respuesta HTTP de la llamada a la API.


```
{
    "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"
}
```

Ejemplo 4: Uso de la utilidad jq para filtrar e imprimir la respuesta que devuelve la API TestState

La TestState API devuelve datos JSON como cadenas de escape en su respuesta. El siguiente AWS CLI ejemplo amplía el ejemplo 3 y utiliza la `jq` utilidad para filtrar e imprimir la respuesta HTTP que devuelve la TestState API en un formato legible para las personas. Para obtener información `jq` y sus instrucciones de instalación, consulte jq on. 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'
```

En el siguiente ejemplo se muestra la salida devuelta en un formato legible para las personas.

```
{
  "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"
}
```

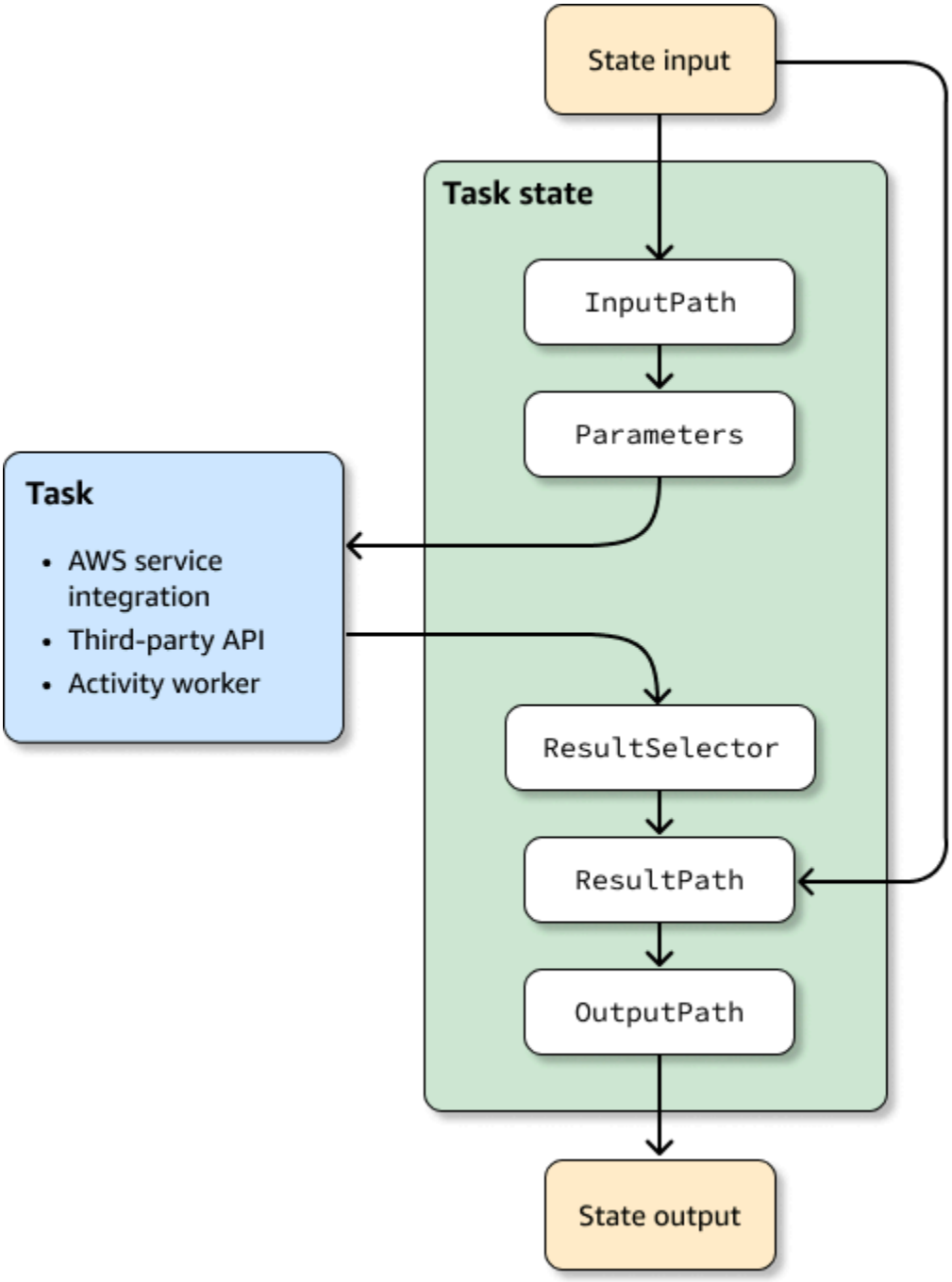
Prueba y depuración del flujo de datos de entrada y de salida

La API `TestState` es útil para probar y depurar los datos que fluyen a través del flujo de trabajo. En esta sección se proporcionan algunos conceptos clave y se explica cómo utilizarlos TestState para este fin.

Conceptos clave

En Step Functions, el proceso de filtrar y manipular datos JSON a medida que pasan por los estados de su máquina de estado se denomina procesamiento de entrada y salida. Para obtener información sobre cómo funciona, consulte Procesamiento de entrada y salida en Step Functions.

Todos los tipos de estado del Lenguaje de estados de Amazon (ASL) (Tarea, Paralelo, Mapa, Aprobado, Esperar, Elección, Correcto y Error) comparten un conjunto de campos comunes para filtrar y manipular los datos JSON que pasan por ellos. Estos campos son: InputPath, Parámetros, ResultSelector, ResultPath y OutputPath. La compatibilidad de cada campo varía según el estado. En tiempo de ejecución, Step Functions aplica cada campo en un orden específico. El siguiente diagrama muestra el orden en el que se aplican estos campos a los datos en un estado Tarea:



La siguiente lista describe el orden de aplicación de los campos de procesamiento de entrada y salida que se muestran en el diagrama.

1. Entrada de estado son los datos JSON pasados al estado actual desde un estado anterior.
2. InputPath filtra una parte de la entrada de estado sin procesar.
3. Parámetros configura el conjunto de valores que se van a pasar a la Tarea.
4. La tarea realiza un trabajo y devuelve un resultado.

5. ResultSelector selecciona un conjunto de valores para excluirlos del resultado de la tarea.
6. ResultPath combina el resultado con la entrada de estado sin procesar o reemplaza el resultado por ella.
7. OutputPath filtra una parte de la salida para pasarla al siguiente estado.
8. Salida de estado son los datos JSON pasados desde estado actual al siguiente estado.

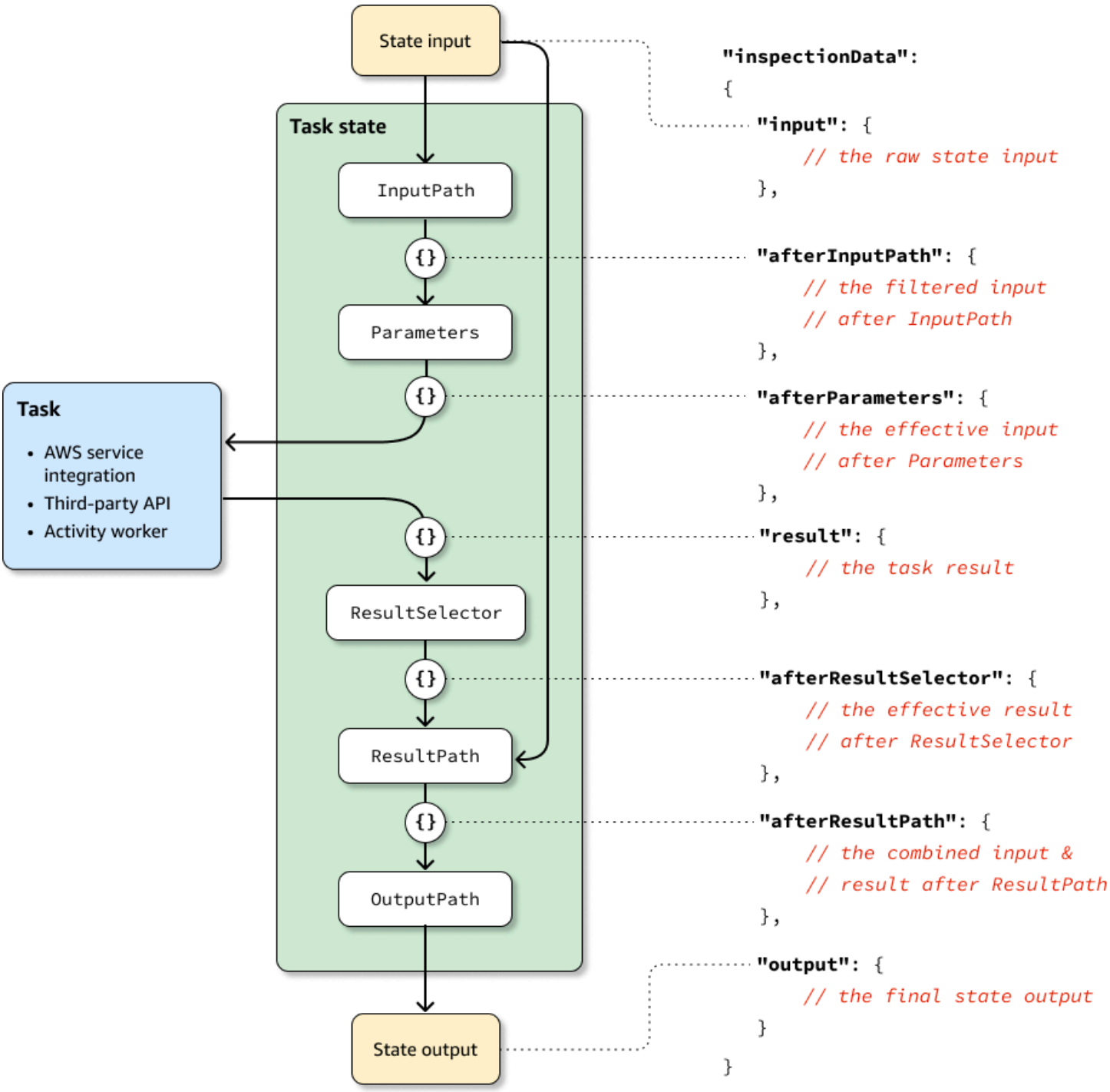
Estos campos de procesamiento de entrada y de salida son opcionales. Si no utiliza ninguno de estos campos en la definición de estado, la tarea consumirá la entrada de estado sin procesar y devolverá el resultado de la tarea como salida de estado.

Se utiliza TestState para inspeccionar el procesamiento de entrada y salida

Cuando llama a la API `TestState` y establece el parámetro `inspectionLevel` en `DEBUG`, la respuesta de la API incluye un objeto llamado `inspectionData`. Este objeto contiene campos que ayudan a inspeccionar cómo se filtraron o manipularon los datos en el estado cuando se ejecutaron. En el ejemplo siguiente se muestra el objeto `inspectionData` para un estado Tarea.

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

En este ejemplo, cada campo que contiene el prefijo `after` muestra los datos después de aplicar un campo concreto. Por ejemplo, `afterInputPath` muestra el efecto de aplicar el campo `InputPath` para filtrar la entrada de estado sin procesar. El siguiente diagrama asigna cada campo de definición de ASL a su campo correspondiente en el objeto `inspectionData`:



Para ver ejemplos del uso de la TestState API para depurar el procesamiento de entrada y salida, consulta lo siguiente:

- Probar un estado mediante el nivel de inspección DEBUG de la consola de Step Functions
- Probar un estado mediante el nivel de inspección DEBUG en el AWS CLI

Probar máquinas de estado de forma local

AWS Step Functions Local es una versión descargable de Step Functions que permite desarrollar y probar aplicaciones utilizando una versión de Step Functions que se ejecuta en su propio entorno de desarrollo. La versión local de Step Functions puede invocar funciones de AWS Lambda que se ejecuten en AWS y localmente. También puede coordinar otros servicios de AWS compatibles.

> **Note**
>
> Step Functions Local funciona con cuentas ficticias.

Mientras ejecuta Step Functions Local puede utilizar una de las siguientes formas de invocar las integraciones de servicios:

- Configurar de puntos de conexión locales para AWS Lambda y otros servicios. Para obtener más información acerca de los puntos de conexión admitidos, consulte Configurar opciones de configuración para Step Functions Local.
- Realizar llamadas directamente a un servicio de AWS desde Step Functions Local.
- Simular la respuesta de integraciones de servicios. Para obtener información acerca del uso de integraciones ficticias de servicios, consulte Uso de integraciones de servicios simuladas.

AWS Step Functions Local está disponible como paquete JAR o como imagen de Docker independiente que se ejecuta en Microsoft Windows, Linux, macOS y otras plataformas compatibles con Java o Docker.

> **Warning**
>
> La versión descargable de AWS Step Functions se ha diseñado solo para su uso en pruebas y no debería utilizarse nunca para procesar información confidencial.

> **Tip**
>
> Asegúrese de utilizar Step Functions Local versión 1.12.0 o superior para poder incluir todas las funciones intrínsecas en sus flujos de trabajo.

En los temas siguientes se explica cómo configurar Step Functions Local mediante Docker y un archivo JAR, y cómo ejecutar Step Functions Local para trabajar con AWS Lambda, AWS Serverless Application Model(AWS SAM) CLI Local u otros servicios compatibles.

Temas

Configurar Step Functions Local (versión descargable) y Docker

La imagen de Docker de Step Functions Local permite empezar a trabajar rápidamente con Step Functions Local mediante una imagen de Docker con todas las dependencias necesarias. La imagen de Docker permite incluir Step Functions Local en las versiones contenedorizadas y como parte de las pruebas continuas de integración.

Para obtener la imagen de Docker para Step Functions Local, vea https://hub.docker.com/r/amazon/aws-stepfunctions-local o escriba el siguiente comando `pull` de Docker.

```
docker pull amazon/aws-stepfunctions-local
```

Para iniciar la versión descargable de Step Functions en Docker, ejecute el siguiente comando `run` de Docker.

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Para interactuar con AWS Lambda u otros servicios admitidos, primero debe configurar sus credenciales y otras opciones de configuración. Para obtener más información, consulte los temas siguientes:

- Configurar opciones de configuración para Step Functions Local
- Credenciales y configuración de Docker

Configurar Step Functions Local (versión descargable) - Versión Java

La versión descargable de AWS Step Functions se proporciona como archivo JAR ejecutable y como imagen de Docker. La aplicación Java se ejecuta en Windows, Linux, macOS y otras plataformas compatibles con Java. Además de Java, debe instalar la AWS Command Line Interface (AWS CLI). Para obtener información sobre cómo instalar y configurar la AWS CLI, consulte la Guía del usuario de la AWS Command Line Interface.

Para configurar y ejecutar Step Functions en su equipo

1. Descargue Step Functions utilizando los siguientes enlaces.

Enlaces de descarga	Suma de comprobación
.tar.gz	.tar.gz.md5
.zip	.zip.md5

2. Extraiga el archivo `.zip`.
3. Pruebe la descarga y consulte la información de la versión.

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (Opcional) Vea una lista de los comandos disponibles.

```
$ java -jar StepFunctionsLocal.jar -h
```

5. Para iniciar Step Functions en el equipo, abra un símbolo del sistema, vaya al directorio donde ha extraído `StepFunctionsLocal.jar` y escriba el comando siguiente.

```
java -jar StepFunctionsLocal.jar
```

6. Para obtener acceso a Step Functions en ejecución local, utilice el parámetro `--endpoint-url`. Por ejemplo, si utiliza la AWS CLI, debería especificar comandos Step Functions como los siguientes:

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> De forma predeterminada, Step Functions Local utiliza una cuenta de prueba y credenciales locales, y la región de AWS se establece en Este de EE. UU. (Norte de Virginia). Para utilizar Step Functions Local con AWS Lambda u otros servicios admitidos, debe configurar sus credenciales y región.
> Si utiliza flujos de trabajo rápidos con Step Functions Local, el historial de ejecución se almacenará en un archivo de registro. No está registrado en CloudWatch Logs. La ruta de acceso del archivo de registro se basará en el ARN del grupo de registro de CloudWatch Logs proporcionado al crear la máquina de estado local. El archivo de registro se almacenará en `/aws/states/log-group-name/${execution_arn}.log` respecto a la ubicación en la que se ejecuta Step Functions Local. Por ejemplo, si el ARN de ejecución es:
>
> ```
> arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-
> wJalrXUtnFEMI
> ```
>
> el archivo de registro será:
>
> ```
> aws/states/log-group-name/arn:aws:states:us-
> east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
> ```

Configurar opciones de configuración para Step Functions Local

Cuando inicia AWS Step Functions Local utilizando el archivo JAR puede definir opciones de configuración mediante la AWS Command Line Interface (AWS CLI) o incluyéndolas en el entorno del sistema. Para Docker, debe especificar estas opciones en un archivo al que haga referencia cuando inicie Step Functions Local.

Opciones de configuración

Cuando configura el contenedor de Step Functions Local para usar un punto de conexión de anulación, como Punto de conexión de Lambda y Punto de conexión de lote, y realiza llamadas a ese

punto de conexión, Step Functions Local no utiliza las credenciales que especifique. La configuración de estas anulaciones de punto de conexión es opcional.

Opción	Línea de comandos	Environment
Cuenta	-account, --aws-account	AWS_ACCOUNT_ID
Region	-region, --aws-region	AWS_DEFAULT_REGION
Escala de tiempo de espera	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Punto de enlace de Lambda	-lambdaEndpoint, --lambda-endpoint	LAMBDA_ENDPOINT
Punto de enlace de Batch	-batchEndpoint, --batch-endpoint	BATCH_ENDPOINT
Punto de enlace de DynamoDB	-dynamoDBEndpoint, --dynamodb-endpoint	DYNAMODB_ENDPOINT
Punto de enlace de ECS	-ecsEndpoint, --ecs-endpoint	ECS_ENDPOINT
Punto de enlace de Glue	-glueEndpoint, --glue-endpoint	GLUE_ENDPOINT
Punto de enlace de SageMaker	-sageMakerEndpoint, --sagemaker-endpoint	SAGE_MAKER_ENDPOINT
Punto de enlace de SQS	-sqsEndpoint, --sqs-endpoint	SQS_ENDPOINT
Punto de enlace de SNS	-snsEndpoint, --sns-endpoint	SNS_ENDPOINT
Punto de conexión de Step Functions	-stepFunctionsEndpoint, --step-functions-endpoint	STEP_FUNCTIONS_ENDPOINT

Credenciales y configuración de Docker

Para configurar Step Functions Local para Docker, cree el archivo siguiente: `aws-stepfunctions-local-credentials.txt`.

Este archivo contiene sus credenciales y otras opciones de configuración. Se puede utilizar lo siguiente como plantilla al crear el archivo `aws-stepfunctions-local-credentials.txt`.

```
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
```

Una vez que haya configurado sus credenciales y las opciones de configuración en `aws-stepfunctions-local-credentials.txt`, inicie Step Functions con el siguiente comando.

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> Se recomienda utilizar el nombre de DNS especial `host.docker.internal`, que se resuelve en la dirección IP interna que utiliza el host, por ejemplo `http://host.docker.internal:8000`. Para obtener más información, consulte la documentación de Docker para Mac y Windows en Networking features in Docker Desktop for Mac y Networking features in Docker Desktop for Windows, respectivamente.

Ejecute Step Functions Local en su ordenador

Utilice la versión local de Step Functions para configurar, desarrollar y probar máquinas de estado en su ordenador.

Ejecute una máquina de estado HelloWorld localmente

Una vez que haya ejecutado Step Functions localmente con la AWS Command Line Interface (AWS CLI), puede iniciar la ejecución de una máquina de estado.

1. Cree una máquina de estado desde la AWS CLI incluyendo la definición de la máquina de estado en una secuencia de escape.

```
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**
>
> El `role-arn` no se utiliza para Step Functions Local, pero debe incluirlo con la sintaxis adecuada. Puede utilizar el nombre de recurso de Amazon (ARN) del ejemplo anterior.

Si ha creado correctamente la máquina de estado, Step Functions responderá con la fecha de creación y el ARN de la máquina de estado.

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. Inicie una ejecución utilizando el ARN de la máquina de estado que ha creado.

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions Local con AWS SAM CLI Local

Puede utilizar la versión local de Step Functions junto con una versión local de AWS Lambda. Para configurar esto, debe instalar y configurar AWS SAM.

Para obtener información acerca de cómo configurar y ejecutar AWS SAM, consulte lo siguiente:

- Configuración de AWS SAM
- Inicie AWS SAM CLI Local.

Una vez que Lambda se esté ejecutando en el sistema local, puede iniciar Step Functions Local. Desde el directorio en el que extrajo los archivos JAR de Step Functions Local, inicie Step Functions Local y utilice el parámetro `--lambda-endpoint` para configurar el punto de conexión de Lambda local.

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

Para obtener más información sobre la ejecución Step Functions Local con AWS Lambda, consulte Prueba de Step Functions y AWS SAM CLI Local.

Prueba de Step Functions y AWS SAM CLI Local

Con AWS Step Functions y AWS Lambda ejecutándose en un equipo local, puede probar la máquina de estado y las funciones de Lambda sin implementar el código en AWS.

Para obtener más información, consulte los siguientes temas:

- Probar máquinas de estado de forma local
- Configuración de AWS SAM

Temas

- Paso 1: Configurar AWS SAM
- Paso 2: Probar AWS SAM CLI Local
- Paso 3: Iniciar AWS SAM CLI Local
- Paso 4: Iniciar Step Functions Local
- Paso 5: Crear una máquina de estado que haga referencia a la función de AWS SAM CLI Local
- Paso 6: Iniciar una ejecución de su máquina de estado local

Paso 1: Configurar AWS SAM

AWS Serverless Application Model (AWS SAM) CLI Local requiere que la AWS Command Line Interface, el AWS SAM y Docker estén instalados.

1. Instalar la CLI de AWS SAM.

   > **Note**
   >
   > Antes de instalar la CLI de AWS SAM, tendrá que instalar la AWS CLI y Docker. Consulte los requisitos previos para instalar la CLI de AWS SAM.

2. Lea la documentación de Inicio rápido de AWS SAM. Asegúrese de seguir los pasos para hacer lo siguiente:

   1. Inicializar la aplicación
   2. Probar la aplicación localmente

   Esto crea un directorio `sam-app` y crea un entorno que incluye una función de Lambda Hello World basada en Python.

Paso 2: Probar AWS SAM CLI Local

Ahora que ha instalado AWS SAM y ha creado la función de Lambda Hello World, puede comprobar que funciona. En el directorio `sam-app`, introduzca el siguiente comando:

```
sam local start-api
```

Se lanzará una instancia local de su función de Lambda. Debería ver una salida similar a esta:

```
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)
```

Abra un navegador y escriba lo siguiente.

```
http://127.0.0.1:3000/hello
```

La salida será una respuesta similar a la siguiente:

```
{"message": "hello world", "location": "72.21.198.66"}
```

Pulse CTRL+C para finalizar la API de Lambda.

Paso 3: Iniciar AWS SAM CLI Local

Ahora que ha probado que la función es correcta, inicie AWS SAM CLI Local. En el directorio `sam-app`, introduzca el siguiente comando:

```
sam local start-lambda
```

Esto inicia AWS SAM CLI Local y proporciona el punto de conexión que se debe utilizar, similar a la siguiente salida:

```
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)
```

Paso 4: Iniciar Step Functions Local

Archivo JAR

Si utiliza la versión del archivo `.jar` de Step Functions Local, inicie Step Functions y especifique el punto de conexión de Lambda. En el directorio donde extrajo los archivos `.jar`, escriba el siguiente comando:

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

Cuando Step Functions Local se inicie, comprueba el entorno y después las credenciales configuradas en su archivo `~/.aws/credentials`. De forma predeterminada, comienza a usar un ID de usuario ficticio y aparece como `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

En caso de que se utilice la versión de Docker de Step Functions Local, lance Step Functions con el siguiente comando:

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Para obtener información sobre cómo instalar la versión de Docker de Step Functions, consulte Configurar Step Functions Local (versión descargable) y Docker.

> **Note**
>
> Puede especificar el punto de conexión con la línea de comandos o configurando variables de entorno si lanza Step Functions desde el archivo `.jar`. Para la versión de Docker, debe especificar los puntos de enlace y las credenciales en un archivo de texto. Consulte Configurar opciones de configuración para Step Functions Local.

Paso 5: Crear una máquina de estado que haga referencia a la función de AWS SAM CLI Local

Una vez que Step Functions Local se esté ejecutando, cree una máquina de estado que haga referencia a la `HelloWorldFunction` que inicializó en Paso 1: Configurar AWS SAM.

```
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"
```

De esta manera, se creará una máquina de estado y se proporcionará un nombre de recurso de Amazon (ARN), que puede utilizar para iniciar una ejecución:

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

Paso 6: Iniciar una ejecución de su máquina de estado local

Una vez que haya creado una máquina de estado, inicie una ejecución. Deberá hacer referencia al punto de conexión y al ARN de la máquina de estado cuando utilice el siguiente comando **`aws stepfunctions`**:

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

Esto inicia una ejecución denominada `test` de su máquina de estado `HelloWorld`.

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

Ahora que Step Functions se ejecuta localmente, puede interactuar con él mediante la AWS CLI. Por ejemplo, para obtener información sobre esta ejecución, utilice el siguiente comando:

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

La llamada a `describe-execution` para una ejecución proporciona detalles más completos, como se indica en la siguiente salida:

```
{
    "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": "{}"
}
```

Uso de integraciones de servicios simuladas

En Step Functions Local, puede probar las rutas de ejecución de sus máquinas de estado sin llamar realmente a los servicios integrados mediante integraciones de servicios simuladas. Para configurar sus máquinas de estado para que utilicen integraciones de servicios simuladas, cree un archivo de configuración simulado. En este archivo, defina la salida deseada de sus integraciones de servicios como respuestas simuladas y las ejecuciones que utilizan sus respuestas simuladas para simular una ruta de ejecución como casos de prueba.

Al proporcionar el archivo de configuración simulado a Step Functions Local, puede probar las llamadas de integración de servicios ejecutando máquinas de estado que utilizan las respuestas simuladas especificadas en los casos de prueba en lugar de realizar llamadas de integración de servicios reales.

> **Note**
>
> Si no especificas respuestas de integración de servicios simuladas en el archivo de configuración simulado, Step Functions Local invocará la integración de AWS servicios mediante el punto final que configuraste al configurar Step Functions Local. Para obtener información sobre la configuración de puntos de conexión para Step Functions Local, consulte Configurar opciones de configuración para Step Functions Local.

Temas

- Conceptos clave de este tema

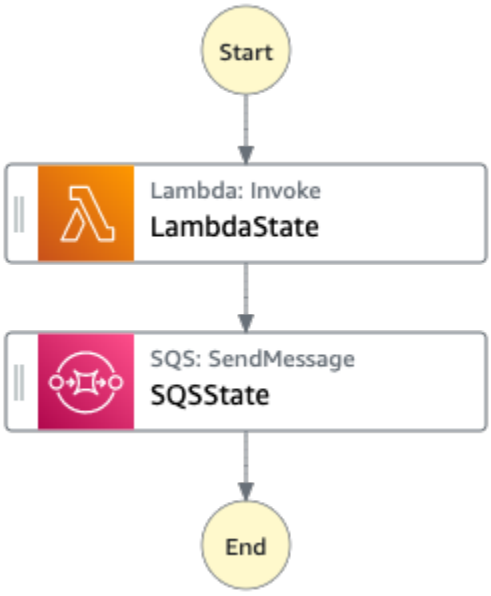
Conceptos clave de este tema

En este tema se utilizan varios conceptos que se definen en la siguiente lista:

- Integraciones de servicios simuladas: se refiere a los estados Task configurados para utilizar respuestas simuladas en lugar de realizar llamadas de servicio reales.
- Respuestas simuladas: se refiere a datos simulados que los estados Task pueden configurar para usar.
- Casos de prueba: se refieren a las ejecuciones de máquinas de estado configuradas para utilizar integraciones de servicios simuladas.
- Archivo de configuración simulado: se refiere al archivo de configuración simulado que contiene JSON, que define las integraciones de servicios simuladas, las respuestas simuladas y los casos de prueba.

Paso 1: Especificar las integraciones de servicios simuladas en un archivo de configuración simulado

Puede probar el AWS SDK de Step Functions y las integraciones de servicios optimizadas mediante Step Functions Local. En la siguiente imagen se muestra la máquina de estado definida en la pestaña Definición de máquina de estado:



Para ello, debe crear un archivo de configuración simulado que contenga las secciones definidas en Presentación de la estructura de la configuración simulada.

1. Cree un archivo con el nombre `MockConfigFile.json` para configurar las pruebas con integraciones de servicios simuladas.

   En el siguiente ejemplo se muestra un archivo de configuración simulado que hace referencia a una máquina de estado con dos estados definidos denominados `LambdaState` y `SQSState`

   Mock configuration file example

   A continuación se muestra un ejemplo de un archivo de configuración simulado que muestra cómo simular respuestas al invocar una función de Lambda y enviar un mensaje a Amazon SQS. En este ejemplo, la máquina de estado `LambdaSQSIntegration` contiene tres casos de prueba denominados `HappyPath`, `RetryPath` y `HybridPath` que simulan los estados `Task` denominados `LambdaState` y `SQSState`. Estos estados utilizan las respuestas de servicio simuladas `MockedLambdaSuccess`, `MockedSQSSuccess` y `MockedLambdaRetry`. Estas respuestas de servicio simuladas se definen en la sección `MockedResponses` del archivo.

   ```
   {
     "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

A continuación se muestra un ejemplo de una definición de máquina de estado denominada `LambdaSQSIntegration`, que define dos estados de tareas de integración de servicios denominados `LambdaState` y `SQSState`. `LambdaState` contiene una política de reintentos basada en `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
    }
  }
}
```

Puede ejecutar la definición de la máquina de estado `LambdaSQSIntegration` a la que se hace referencia en el archivo de configuración simulado mediante uno de los siguientes casos de prueba:

- `HappyPath`: esta prueba simula la salida de `LambdaState` y `SQSState` mediante `MockedLambdaSuccess` y `MockedSQSSuccess`, respectivamente.
  - El `LambdaState` devolverá los siguientes valores:

```
"0":{
  "Return":{
    "StatusCode":200,
    "Payload":{
      "StatusCode":200,
      "body":"Hello from Lambda!"
    }
  }
}
```

- El `SQSState` devolverá los siguientes valores:

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- `RetryPath`: esta prueba simula la salida de `LambdaState` y `SQSState` mediante `MockedLambdaRetry` y `MockedSQSSuccess`, respectivamente. Además, `LambdaState` está configurada para realizar cuatro reintentos. Las respuestas simuladas para estos intentos se definen e indexan en el estado `MockedLambdaRetry`.
  - El intento inicial finaliza con un error en la tarea que contiene un mensaje de causa y error, como se muestra en el siguiente ejemplo:

```
"0":{
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
}
```

  - El primer y el segundo intento finalizan con un error en la tarea que contiene un mensaje de causa y error, como se muestra en el siguiente ejemplo:

```
"1-2":{
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

  - El tercer intento finaliza con una tarea exitosa que contiene el resultado de estado de la sección Carga en la respuesta de Lambda simulada.

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
```

```
      "body": "Hello from Lambda!"
    }
  }
}
```

> **Note**
>
> - En el caso de los estados con una política de reintentos, Step Functions Local agotará los reintentos establecidos en la política hasta que reciba una respuesta correcta. Esto significa que debe denotar simulaciones para los reintentos con números de intentos consecutivos y debe cubrir todos los reintentos antes de obtener una respuesta correcta.
> - Si no especifica una respuesta simulada para un reintento específico (por ejemplo, si reintenta «3»), la ejecución de la máquina de estado generará un error.

- `HybridPath`: esta prueba simula la salida de `LambdaState`. Después de que `LambdaState` se ejecute correctamente y reciba datos simulados como respuesta, `SQSState` realiza una llamada de servicio real al recurso especificado en producción.

Para obtener información sobre cómo iniciar las ejecuciones de pruebas con integraciones de servicios simuladas, consulte Paso 3: Ejecutar pruebas de integración de servicios simuladas.

2. Asegúrese de que la estructura de las respuestas simuladas se ajuste a la estructura de las respuestas de servicio reales que recibe cuando realiza llamadas de servicio integradas. Para obtener información sobre los requisitos estructurales de las respuestas simuladas, consulte Configuración de integraciones de servicios simuladas.

   En el archivo de configuración simulada del ejemplo anterior, las respuestas simuladas definidas en `MockedLambdaSuccess` y `MockedLambdaRetry` se ajustan a la estructura de las respuestas reales que se devuelven al llamar a `HelloFromLambda`.

   > **Important**
   >
   > Las respuestas del servicio de AWS pueden variar entre los distintos servicios. Step Functions Local no valida si las estructuras de respuesta simuladas se ajustan a las estructuras de respuesta del servicio reales. Debe asegurarse de que sus respuestas simuladas se ajusten a las respuestas reales antes de realizar la prueba. Para revisar la

estructura de las respuestas de servicios, puede realizar las llamadas de servicio reales mediante Step Functions o consultar la documentación de dichos servicios.

Paso 2: Proporcionar el archivo de configuración simulado a Step Functions Local

Puede proporcionar el archivo de configuración simulado a Step Functions Local de una de las siguientes maneras:

Docker

> **Note**
>
> Si utiliza la versión Docker de Step Functions Local, puede proporcionar el archivo de configuración simulado utilizando únicamente una variable de entorno. Además, debe montar el archivo de configuración simulado en el contenedor de Step Functions Local en el arranque inicial del servidor.

Monte el archivo de configuración simulado en cualquier directorio del contenedor de Step Functions Local. A continuación, defina una variable de entorno denominada `SFN_MOCK_CONFIG` que contenga la ruta al archivo de configuración simulado del contenedor. Este método permite que el archivo de configuración simulado reciba cualquier nombre siempre que la variable de entorno contenga la ruta y el nombre del archivo.

El siguiente comando muestra el formato para iniciar la imagen de Docker.

```
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
```

En el siguiente ejemplo se utiliza el comando para iniciar la imagen de Docker.

```
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

Use una de las siguientes formas para proporcionar el archivo de configuración simulado a Step Functions Local:

- Coloque el archivo de configuración simulado en el mismo directorio que `Step FunctionsLocal.jar`. Al usar este método, debe asignar un nombre al archivo de configuración simulado `MockConfigFile.json`.
- En la sesión en la que se ejecuta Step Functions Local, defina una variable de entorno denominada `SFN_MOCK_CONFIG`, con la ruta completa del archivo de configuración simulado. Este método permite que el archivo de configuración simulado reciba cualquier nombre siempre que la variable de entorno contenga la ruta y el nombre del archivo. En el siguiente ejemplo, la variable `SFN_MOCK_CONFIG` se establece para que apunte a un archivo de configuración simulado denominado `EnvSpecifiedMockConfig.json`, ubicado en el directorio `/home/workspace`.

```
export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
```

> **Note**
>
> - Si no se proporciona la variable de entorno `SFN_MOCK_CONFIG` a Step Functions Local, de forma predeterminada, intentará leer un archivo de configuración simulado denominado `MockConfigFile.json` en el directorio desde el que se inició Step Functions Local.
> - Si coloca el archivo de configuración simulado en el mismo directorio que `Step FunctionsLocal.jar` y establece la variable de entorno `SFN_MOCK_CONFIG`, Step Functions Local leerá el archivo especificado por la variable de entorno.

Paso 3: Ejecutar pruebas de integración de servicios simuladas

Después de crear y proporcionar un archivo de configuración simulado a Step Functions Local, ejecute la máquina de estado configurada en el archivo de configuración simulado mediante integraciones de servicios simuladas. A continuación, compruebe los resultados de la ejecución mediante una acción de API.

1. Cree una máquina de estado basada en la definición mencionada anteriormente en el archivo de configuración simulado.

```
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. Ejecute la máquina de estado mediante integraciones de servicios simuladas.

   Para usar el archivo de configuración simulado, realice una llamada a la API `StartExecution` en una máquina de estado configurada en el archivo de configuración simulado. Para ello, añada el sufijo ,#*`test_name`*, al ARN de la máquina de estado utilizado por `StartExecution`. *`test_name`* es un caso de prueba, que se configura para la máquina de estado en el mismo archivo de configuración simulado.

   El siguiente comando es un ejemplo que utiliza la máquina de estado `LambdaSQSIntegration` y la configuración simulada. En este ejemplo, la máquina de estado `LambdaSQSIntegration` se ejecuta mediante la prueba `HappyPath` definida en Paso 1: Especificar las integraciones de servicios simuladas en un archivo de configuración simulado. La prueba `HappyPath` contiene la configuración de la ejecución para gestionar las llamadas de integración de servicios simuladas que los estados `LambdaState` y `SQSState` realizan mediante las respuestas de servicio simuladas `MockedLambdaSuccess` y `MockedSQSSuccess`.

```
aws stepfunctions start-execution \
    --endpoint http://localhost:8083 \
    --name executionWithHappyPathMockedServices \
    --state-machine arn:aws:states:us-
east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
```

3. Vea el estado de la respuesta de ejecución de la máquina.

La respuesta a una llamada a `StartExecution` mediante una prueba de integración de servicios simulada es la misma que la respuesta a una llamada a `StartExecution` normal, que devuelve el ARN de ejecución y la fecha de inicio.

A continuación se muestra un ejemplo de respuesta a una llamada `StartExecution` mediante la prueba de integración de servicios simulada:

```
{
   "startDate":"2022-01-28T15:03:16.981000-05:00",
   "executionArn":"arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
}
```

4. Compruebe los resultados de la ejecución realizando una llamada a la API `ListExecutions`, `DescribeExecution` o `GetExecutionHistory`

```
aws stepfunctions get-execution-history \
    --endpoint http://localhost:8083 \
    --execution-arn arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
```

En el siguiente ejemplo se muestran partes de una respuesta a una llamada a `GetExecutionHistory` mediante el ARN de ejecución de la respuesta de ejemplo que se muestra en el paso 2. En este ejemplo, la salida de `LambdaState` y `SQSState` son los datos simulados definidos en `MockedLambdaSuccess` y `MockedSQSSuccess` en el archivo de configuración simulado. Además, los datos simulados se utilizan de la misma manera que los datos devueltos al realizar llamadas de integración de servicios reales. Además, en este ejemplo, la salida de `LambdaState` se transfiere a `SQSState` como entrada.

```
{
    "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
                }
            }
        },
        ...
    ]
}
```

Archivo de configuración para integraciones de servicios simuladas

Para usar integraciones de servicios simuladas, primero debe crear un archivo de configuración simulado denominado `MockConfigFile.json` que contenga las configuraciones simuladas. A continuación, proporcione a Step Functions Local el archivo de configuración simulado. Este archivo de configuración define los casos de prueba, que contienen estados simulados que utilizan respuestas de integración de servicios simuladas. La siguiente sección contiene información sobre la estructura de la configuración simulada, que incluye los estados simulados y las respuestas simuladas:

Temas

- Presentación de la estructura de la configuración simulada
- Configuración de integraciones de servicios simuladas

Presentación de la estructura de la configuración simulada

Una configuración simulada es un objeto JSON que contiene los siguientes campos de nivel superior:

- `StateMachines`: los campos de este objeto representan máquinas de estado configuradas para utilizar integraciones de servicios simuladas.
- `MockedResponse`: los campos de este objeto representan respuestas simuladas para las llamadas de integración de servicios.

A continuación se muestra un ejemplo de un archivo de configuración simulado que incluye una definición `StateMachine` y `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!"
          }
        }
      }
    }
  }
}
```

Referencia de campo de configuración simulada

En las siguientes secciones se explican los campos de objetos de nivel superior que debe definir en la configuración simulada.

- StateMachines
- MockedResponses

StateMachines

El objeto `StateMachines` define qué máquinas de estado utilizarán integraciones de servicios simuladas. La configuración de cada máquina de estado se representa como un campo de nivel superior de `StateMachines`. El nombre del campo es el nombre de la máquina de estado y el valor es un objeto que contiene un único campo denominado `TestCases`, cuyos campos representan casos de prueba de esa máquina de estado.

La siguiente sintaxis muestra una máquina de estado con dos casos de prueba:

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

Los campos de `TestCases` representan casos de prueba individuales para la máquina de estado. El nombre de cada caso de prueba debe ser único para cada máquina de estado y el valor de cada caso de prueba es un objeto que especifica una respuesta simulada para utilizarla en los estados Task de la máquina de estado.

El siguiente ejemplo de `TestCase` vincula dos estados `Task` a dos `MockedResponses`:

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
  "AnotherTaskState": "AnotherMockedResponse"
}
```

MockedResponses

`MockedResponses` es un objeto que contiene varios objetos de respuesta simulados con nombres de campo únicos. Un objeto de respuesta simulado define el resultado correcto o la salida con error de cada invocación de un estado Task simulado. El número de invocación se especifica mediante cadenas de números enteros individuales, como «0», «1», «2» y «3», o un rango inclusivo de números enteros, como «0-1», «2-3».

Cuando se simula una tarea, se debe especificar una respuesta simulada para cada invocación. La respuesta debe contener un único campo denominado `Return` o `Throw` cuyo valor sea el resultado o error de la invocación de la tarea simulada. Si no especifica una respuesta simulada, la ejecución de la máquina de estado generará un error.

A continuación se muestra un ejemplo de una `MockedResponse` con `Throw` y `Return` objetos. En este ejemplo, las tres primeras veces que se ejecuta la máquina de estado, se devuelve la

respuesta especificada en `"0-2"`, y la cuarta vez que se ejecuta la máquina de estado, se devuelve la respuesta especificada en `"3"`.

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> Si utiliza un estado `Map` y quiere garantizar respuestas predecibles para el estado `Map`, defina el valor de `maxConcurrency` en 1. Si establece un valor superior a 1, Step Functions Local ejecutará varias iteraciones simultáneamente, lo que provocará que el orden general de ejecución de los estados de las iteraciones sea impredecible. Además, esto puede provocar que Step Functions Local utilice diferentes respuestas simuladas para los estados de iteración de una ejecución a la siguiente.

Return

`Return` se representa como un campo de los objetos `MockedResponse`. Especifica el resultado correcto de un estado Task simulado.

A continuación se muestra un ejemplo de un objeto `Return` que contiene una respuesta simulada para llamar a `Invoke` en una función de Lambda:

```
"Return": {
  "StatusCode": 200,
  "Payload": {
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
```

```
}
```

Throw

`Throw` se representa como un campo de los objetos `MockedResponse`. Especifica la salida de error de una tarea fallida. El valor de `Throw` debe ser un objeto que contenga campos `Error` y `Cause` con valores de cadena. Además, el valor de cadena que especifique en el campo `Error` en `MockConfigFile.json` debe coincidir con los errores gestionados en las secciones `Retry` y `Catch` de su máquina de estado.

A continuación se muestra un ejemplo de un objeto `Throw` que contiene una respuesta simulada para llamar a `Invoke` en una función de Lambda:

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
}
```

Configuración de integraciones de servicios simuladas

Puede simular cualquier integración de servicios mediante Step Functions Local. Sin embargo, Step Functions Local no exige que las simulaciones sean las mismas que las API reales. Una tarea simulada nunca llamará al punto de conexión del servicio. Si no especifica una respuesta simulada, una tarea intentará llamar a los puntos de conexión del servicio. Además, Step Functions Local generará automáticamente un token de tarea cuando simule una tarea con `.waitForTaskToken`.

Prácticas recomendadas para Step Functions

Las siguientes prácticas recomendadas para implementar flujos de trabajo de AWS Step Functions pueden ayudarle a optimizar el rendimiento de las implementaciones.

Temas

Utilizar tiempos de espera para evitar las ejecuciones bloqueadas

De forma predeterminada, Amazon States Language no especifica tiempos de espera para las definiciones de máquina de estado. Sin un tiempo de espera explícito, Step Functions a menudo se basa únicamente en una respuesta de un proceso de trabajo empleado de actividad para saber que una tarea se ha completado. Si algo va mal y no se especifica el campo `TimeoutSeconds` para un estado `Activity` o `Task`, una ejecución se bloquea a la espera una respuesta que nunca llegará.

Para evitar esto, especifique un límite de tiempo de espera razonable cuando cree una `Task` en la máquina de estado. Por ejemplo:

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

Si utiliza una devolución de llamada con un token de tarea (.waitForTaskToken), le recomendamos que utilice latidos y añada el campo `HeartbeatSeconds` a su definición de estado de `Task`. Puede

configurar `HeartbeatSeconds` para que sea inferior al tiempo de espera de la tarea, de modo que si el flujo de trabajo falla debido a un error de latido, sabrá que se debe a un error en la tarea y no a que la tarea tarde mucho tiempo en completarse.

```
{
  "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
    }
  }
}
```

Para obtener más información, consulte Estado de la tarea en la documentación de Amazon States Language.

> **Note**
>
> Puede establecer un tiempo de espera para la máquina de estado mediante el campo `TimeoutSeconds` de su definición de Amazon States Language. Para obtener más información, consulte Estructura de las máquinas de estado.

Utilizar los ARN de Amazon S3 en lugar de pasar cargas de gran tamaño

Las ejecuciones que pasan cargas de gran tamaño entre los estados pueden terminarse. Si los datos que va a transferir entre estados pueden superar los 256 KB, utilice Amazon Simple Storage Service (Amazon S3) para almacenar los datos y analice el Nombre de recurso de Amazon (ARN) del bucket en el parámetro `Payload` para obtener el nombre del bucket y el valor de clave. También puede ajustar la implementación para que se pasen cargas más pequeñas en las ejecuciones.

En el siguiente ejemplo, una máquina de estados pasa la entrada a una función AWS Lambda, que procesa un archivo JSON en un bucket de Amazon S3. Tras ejecutar esta máquina de estados, la función de Lambda lee el contenido del archivo JSON y devuelve el contenido del archivo como salida.

Crear la función de Lambda

La siguiente función de Lambda denominada *pass-large-payload* lee el contenido de un archivo JSON almacenado en un bucket de Amazon S3 específico.

> **Note**
>
> Tras crear esta función de Lambda, no olvide proporcionar a su rol de IAM el permiso adecuado para leer desde un bucket de Amazon S3. Por ejemplo, asocie el permiso Amazons3ReadOnlyAccess al rol de la función de Lambda.

```
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
```

Crear la máquina de estado

La siguiente máquina de estados invoca la función de Lambda que creó anteriormente.

```
{
   "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
      }
   }
}
```

En lugar de pasar una cantidad de datos grande en la entrada, podría guardar esos datos en un bucket de Amazon S3 y pasar el Nombre de recurso de Amazon (ARN) y del bucket en el parámetro `Payload` para obtener el nombre del bucket y el valor de clave. Entonces, la función de Lambda puede usar ese ARN para obtener acceso a los datos directamente. A continuación se muestra una entrada de ejemplo para la ejecución de la máquina de estado, donde los datos se almacenan en `data.json` en un bucket de Amazon S3 llamado *`large-payload-json`*.

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

Evitar alcanzar la cuota de historial

AWS Step Functions tiene una cuota rígida de 25 000 entradas en el historial de ejecución. Cuando una ejecución alcanza los 24 999 eventos, espera a que se produzca el siguiente evento.

- Si el evento número 25 000 es `ExecutionSucceeded`, la ejecución finaliza correctamente.
- Si el evento número 25 000 no es `ExecutionSucceeded`, se registra el evento `ExecutionFailed` y la ejecución de la máquina de estado produce un error al alcanzar el límite del historial

Para evitar alcanzar esta cuota para las ejecuciones de larga duración, pruebe alguna de las soluciones alternativas siguientes:

- Utilice el estado Map en modo distribuido. En este modo, el estado `Map` ejecuta cada iteración como una ejecución de flujo de trabajo secundario, lo que permite una alta simultaneidad de hasta 10 000 ejecuciones de flujos de trabajo secundarios en paralelo. Cada ejecución de flujo de trabajo secundario tiene su propio historial de ejecución independiente del flujo de trabajo principal.
- Inicie una nueva ejecución de máquina de estado directamente desde el estado `Task` de una ejecución en curso. Para iniciar dichas ejecuciones de flujos de trabajo anidados, utilice la acción de la API `StartExecution` de Step Functions en la máquina de estado principal junto con los parámetros necesarios. Para obtener más información sobre el uso de flujos de trabajo anidados, consulte Iniciar ejecuciones de flujo de trabajo desde un estado de tarea o el tutorial Utilizar una acción de la API Step Functions para continuar una nueva ejecución.

  > **Tip**
  >
  > Para implementar un ejemplo de un flujo de trabajo anidadoCuenta de AWS, consulte el Módulo 13: Flujos de trabajo rápidos anidados.

- Implemente un patrón que use una función de AWS Lambda que pueda comenzar una nueva ejecución de la máquina de estado para dividir el trabajo en curso en varias ejecuciones de flujo de trabajo. Para obtener más información, consulte el tutorial Uso de una función de Lambda para continuar con una nueva ejecución.

Gestionar excepciones de servicio de Lambda

En ocasiones, AWS Lambda puede experimentar errores de servicio transitorios. En este caso, la invocación de resultados de Lambda da como resultado un error 500, como `ClientExecutionTimeoutException`, `ServiceException`, `AWSLambdaException` o `SdkClientException`. Use la práctica recomendada de controlar estas excepciones de manera

proactiva en la máquina de estado y ejecutar `Retry` para volver a invocar la función de Lambda o `Catch` para capturar el error.

Los errores de Lambda se notifican como `Lambda.`*`ErrorName`*. Para reintentar la función de Lambda después de una excepción de error de servicio, puede usar el código `Retry` siguiente:

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Los errores no controlados en Lambda se notifican como `Lambda.Unknown` en el resultado del error. Estos incluyen los errores de memoria insuficiente y los tiempos de espera de funciones. Puede buscar coincidencias de estos errores con `Lambda.Unknown`, `States.ALL` o `States.TaskFailed` para controlarlos. Cuando Lambda alcanza el número máximo de invocaciones, el error es `Lambda.TooManyRequestsException`. Para obtener más información acerca de errores de función de Lambda, consulte Tratamiento de errores y reintentos automáticos en la Guía para desarrolladores de AWS Lambda.

Para obtener más información, consulte los siguientes temas:

- Reintento después de un error
- Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos
- Errores de Invoke de Lambda

Evitar la latencia al sondear tareas de actividad

La API de `GetActivityTask` se ha diseñado para proporcionar un `taskToken` exactamente una vez. Si se descarta un `taskToken` durante la comunicación con un proceso de trabajo de actividad, se puede bloquear una serie de solicitudes `GetActivityTask` durante 60 segundos al esperar una respuesta hasta que se agote el tiempo de espera de `GetActivityTask`.

Si solo tiene un pequeño número de sondeos que esperan una respuesta, es posible que todas las solicitudes se pongan en cola detrás de la solicitud bloqueada y se detengan. Sin embargo, si tiene un gran número de sondeos pendientes para cada Nombre de recurso de Amazon (ARN) de actividad y algún porcentaje de sus solicitudes están bloqueadas esperando, habrá muchas más que todavía pueden obtener un `taskToken` y comenzar a procesar el trabajo.

Para los sistemas de producción recomendamos al menos 100 sondeos abiertos por ARN de actividad en cada momento. Si se bloquea un sondeo y una parte de los sondeos se ponen en cola detrás de ella, todavía hay muchas más solicitudes que recibirán un `taskToken` para procesar el trabajo mientras la solicitud `GetActivityTask` está bloqueada.

Para evitar este tipo de problemas de latencia al sondear las tareas:

- Implemente los sondeos como subprocesos independientes del trabajo en la implementación del proceso de trabajo de actividad.
- Tenga al menos 100 sondeos abiertos por ARN de actividad en cada momento.

> **Note**
>
> El escalado hasta 100 sondeos abiertos por ARN puede resultar costoso. Por ejemplo, 100 sondeos de funciones de Lambda por ARN resultan 100 veces más costosos que tener una sola función de Lambda con 100 subprocesos de sondeo. Tanto para reducir la latencia como para minimizar el costo, use un lenguaje que tenga E/S asincrónica e implemente varios subprocesos de sondeo por trabajador. Para ver un ejemplo de un proceso de actividad en el que los subprocesos de sondeo están separados de los subprocesos de trabajo, consulte Ejemplo de proceso de trabajo de actividad en Ruby.

Para obtener más información sobre las actividades y los procesos de trabajo de actividad, consulte Actividades.

Selección de flujos de trabajo estándar o rápidos

AWS Step Functions ofrece flujos de trabajo estándar como tipo de flujo de trabajo predeterminado, con la opción de elegir flujos de trabajo rápidos.

Puede elegir flujos de trabajo estándar cuando necesite flujos de trabajo de larga duración, duraderos y auditables, o flujos de trabajo rápidos para cargas de trabajo de procesamiento de

eventos de gran volumen. Las ejecuciones de la máquina de estado se comportarán de forma diferente en función del `Type` que seleccione. El `Type` que seleccione no se podrá cambiar después de que la máquina de estado se haya creado.

- Para obtener información detallada acerca de las diferencias entre los flujos de trabajo estándar y rápidos, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.
- Para obtener información sobre cómo optimizar los costes al crear flujos de trabajo sin servidor mediante Step Functions, consulte Optimización de costes mediante flujos de trabajo rápidos.

Restricciones de tamaño de política de recursos de registros de Amazon CloudWatch

Las políticas de recursos de Registros de Amazon CloudWatch están limitadas a 5120 caracteres. Cuando Registros de Amazon CloudWatch detecta que una política se acerca a este límite de tamaño, habilita automáticamente los grupos de registro que comienzan con `/aws/vendedlogs/`.

Cuando cree una máquina de estado con el registro habilitado, Step Functions debe actualizar la política de recursos de Registros de Amazon CloudWatch con el grupo de registro que especifique. Para evitar alcanzar el límite de tamaño de la política de recursos de Registros de Amazon CloudWatch, ponga el prefijo a los nombres del grupo de registro con `/aws/vendedlogs/`. Al crear un grupo de registros en la consola de Step Functions, a los nombres de los grupos de registro se les añade el prefijo `/aws/vendedlogs/states`. Para obtener más información, consulte Habilitar el registro desde determinados servicios de AWS.

Si no puede acceder a CloudWatch Logs, Unable to access the CloudWatch Logs consulte para obtener más información.

Uso AWS Step Functions con otros servicios

Obtén información sobre cómo coordinar otras API Servicios de AWS o cómo llamar a ellas de terceros. AWS Step Functions

Temas

Llama a otros AWS servicios

Con las integraciones de AWS servicios, puedes activar las acciones de la API y coordinar las ejecuciones directamente desde tu flujo de trabajo. Puedes usar las integraciones del AWS SDK de Step Functions para llamar a cualquiera de los más de doscientos AWS servicios directamente desde tu máquina de estados, lo que te da acceso a más de nueve mil acciones de API. También puede utilizar las integraciones optimizadas de Step Functions, cada una de las cuales se ha personalizado para proporcionar una funcionalidad especial para el flujo de trabajo. Algunas acciones de la API están disponibles en ambos tipos de integración. Siempre que sea posible, te recomendamos que utilices la integración optimizada.

Coordine estos servicios directamente desde un estado `Task` en Amazon States Language. Por ejemplo, con Step Functions, puede llamar a otros servicios para:

- Invoca cualquier AWS Lambda función.
- Ejecute un AWS Batch trabajo y, a continuación, realice diferentes acciones en función de los resultados.
- Insertar u obtener un elemento de Amazon DynamoDB.
- Ejecutar una tarea de Amazon Elastic Container Service (Amazon ECS) y esperar a que finalice.
- Publicar en un tema en Amazon Simple Notification Service (Amazon SNS).

- Enviar un mensaje en Amazon Simple Queue Service (Amazon SQS).
- Gestiona un trabajo para AWS Glue Amazon SageMaker.
- Genere flujos de trabajo para ejecutar trabajos de Amazon EMR.
- Lanza una ejecución AWS Step Functions de flujo de trabajo.

Integraciones optimizadas

Step Functions ha personalizado las integraciones optimizadas para brindar una funcionalidad especial para un contexto de flujo de trabajo. Por ejemplo, Lambda `Invoke` convierte la salida de la API de un JSON de escape en un objeto JSON. AWS Batch`SubmitJob` permite pausar la ejecución hasta que se complete el trabajo. El primer conjunto de integraciones optimizadas se lanzó en 2018 y ahora hay más de cincuenta API.

AWS Integraciones de SDK

AWS Las integraciones del SDK funcionan exactamente igual que una llamada a la API estándar con el AWS SDK. Ofrecen la posibilidad de llamar a más de nueve mil API de los más de doscientos AWS servicios directamente desde la definición automática de su estado.

Compatibilidad con patrones de integración

Los flujos de trabajo estándar y los flujos de trabajo rápidos admiten las mismas integraciones, pero no los mismos patrones de integración.

- La compatibilidad con los patrones de integración es diferente para cada integración.
- Los flujos de trabajo de Express no admiten Run a Job (.sync) ni Wait for Callback (). waitForTaskSímbolo).
- Para obtener más información, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

Standard Workflows

Integraciones de servicios compatibles

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
Integraciones optimizadas	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	✓		✓

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**.sync**)	Espere la devolución de la llamada (**.waitForTaskToken**)
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integraciones de SDK	Más de doscientas	✓		✓

Express Workflows

Integraciones de servicios compatibles

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**.sync**)	Espere la devolución de la llamada (**.waitForTaskToken**)
Integraciones	Amazon API Gateway	✓		
	Amazon Athena	✓		

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
optimizadas	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	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**.sync**)	Espere la devolución de la llamada (**.waitForTaskToken**)
	AWS Step Functions	✓		
AWS Integraciones de SDK	Más de doscientas	✓		

Acceso entre cuentas

Step Functions proporciona acceso multicuenta a los recursos configurados Cuentas de AWS en diferentes flujos de trabajo. Con las integraciones de servicios de Step Functions, puedes invocar cualquier AWS recurso multicuenta, incluso si Servicio de AWS no es compatible con políticas basadas en recursos o llamadas entre cuentas.

Para obtener más información, consulte Acceder a los recursos de otras partes Cuentas de AWS de sus flujos de trabajo.

AWS Integraciones de servicios SDK

AWS Step Functions se integra con Servicios de AWS, lo que te permite llamar a las acciones de la API de cada servicio desde tu flujo de trabajo. Puedes usar las integraciones del AWS SDK de Step Functions para llamar a casi cualquier acción Servicio de AWS de la API desde tu máquina de estados. También puede utilizar las integraciones optimizadas de Step Functions, cada una de las cuales se ha personalizado para proporcionar una funcionalidad especial para el flujo de trabajo.

Es posible que algunos servicios o SDK no estén disponibles como integraciones de AWS SDK, ya sea de forma temporal o permanente. Es posible que las interacciones del SDK no estén disponibles en los servicios lanzados recientemente hasta una actualización posterior. Algunos servicios requieren una configuración personalizada, como especificar un punto de conexión específico para

el cliente, lo que puede ser más adecuado para una integración optimizada. Otros SDK no son adecuados para su uso en un flujo de trabajo, como los que se utilizan para la transmisión de audio o vídeo. Por último, algunos servicios pueden retenerse hasta que superen determinadas validaciones internas realizadas por Step Functions.

> **Tip**
>
> Para implementar un ejemplo de un flujo de trabajo que utilice integraciones de AWS SDK Cuenta de AWS, consulte el Módulo 9: Integraciones de servicios de AWS SDK en The Workshop. AWS Step Functions

Temas

Uso de integraciones de servicios AWS de SDK

Para usar las integraciones AWS del SDK, debe especificar el nombre del servicio y la llamada a la API y, si lo desea, un patrón de integración del servicio.

> **Note**
>
> - Los parámetros Step Functions se expresan en PascalCase, incluso si la API de servicio nativa está en CamelCase. Por ejemplo, podría usar la acción de la API de Step Functions `startSyncExecution` y especificar su parámetro como `StateMachineArn`.
> - Para las acciones de API que aceptan parámetros enumerados, como la acción de API `DescribeLaunchTemplateVersions` para Amazon EC2, especifique una versión plural del nombre del parámetro. Por ejemplo, especifique `Filters` para el parámetro `Filter.N` de la acción de la API `DescribeLaunchTemplateVersions`.

Puedes llamar a los servicios del AWS SDK directamente desde el idioma de Amazon States en el `Resource` campo del estado de una tarea. Para ello, utilice la siguiente sintaxis:

`arn:aws:states:::aws-sdk:serviceName:apiAction.[serviceIntegrationPattern]`

Por ejemplo, para Amazon EC2, podría utilizar `arn:aws:states:::aws-sdk:ec2:describeInstances`. Esto devolvería la salida como se define para la llamada a la API describeInstances de Amazon EC2.

Si se produce un error en una integración del AWS SDK, el campo Error resultante estará compuesto por el nombre del servicio y el nombre del error, separados por un punto:*ServiceName*.*ErrorName*. Tanto el nombre del servicio como el nombre del error están en Pascal case. También puede ver el nombre del servicio, en minúsculas, en el campo Resource del estado Task. Puedes encontrar los nombres de los posibles errores en la documentación de referencia de la API del servicio de destino.

Por ejemplo, puedes usar la integración del `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK. La referencia de la API de AWS Private Certificate Authority indica que la acción de la API `DeleteCertificateAuthority` puede provocar `ResourceNotFoundException`, por ejemplo. Para gestionar este error, debe especificar el valor de `AcmPca.ResourceNotFoundException` del Error en los “reintentadores” o detectores del estado de la tarea. Para obtener más información sobre la gestión de errores en Step Functions, consulte Control de errores en Step Functions.

Step Functions no puede generar automáticamente políticas de IAM para las integraciones del AWS SDK. Tras crear la máquina de estado, tendrá que ir a la consola de IAM y configurar las políticas de rol. Para obtener más información, consulte Políticas de IAM para servicios integrados.

Consulta el Recopile información sobre los buckets de Amazon S3 mediante integraciones de servicios de AWS SDK tutorial para ver un ejemplo de cómo usar AWS las integraciones del SDK.

Integraciones de servicios AWS de SDK compatibles

En la siguiente tabla se enumeran las integraciones de servicios del AWS SDK compatibles con Step Functions. La columna de Recursos del estado *Task* muestra la sintaxis para llamar a una acción de API específica cuando se utiliza la integración para el servicio especificado en la columna Nombre de servicio de la izquierda. La columna Fecha de admisión proporciona información sobre las fechas en las que se permitió la integración del servicio. Además, la columna Prefijo de excepción de la derecha muestra los prefijos de excepción para cada integración de servicios. Estos prefijos de excepción están presentes en las excepciones que se generan al realizar por error una integración del servicio AWS SDK con Step Functions.

Note

- Los servicios marcados con *** tienen acciones de API que Step Functions no admite en este momento. Para obtener información sobre las acciones que no se admiten en un servicio, consulte la tabla Acciones de API no admitidas para servicios admitidos.
- Para obtener información sobre las actualizaciones realizadas con cada lanzamiento para ampliar la compatibilidad con las integraciones del AWS SDK, consulte. Registro de cambios para las integraciones AWS de SDK compatibles

Important

El soporte para las acciones de la API se publica trimestralmente. Es posible que las actualizaciones de las acciones ya compatibles, como los nuevos parámetros, no estén disponibles de forma inmediata.

Nombre del servicio	Recurso de estado **`Task`**	Fecha de admisión	Prefijo de excepción
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric: [apiAction]`	18 de enero de 2024	AppFabric
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi: [apiAction]`	18 de enero de 2024	B2Bi
Exportaciones de datos de AWS	`arn:aws:states:::aws-sdk:bcmdataexports: [apiAction]`	18 de enero de 2024	BcmDataExports
Amazon Bedrock	`arn:aws:states:::aws-sdk:bedrock: [apiAction]`	18 de enero de 2024	Bedrock

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Bedrock Agents	`arn:aws:states:::aws-sdk:bedrockagent: [apiAction]`	18 de enero de 2024	BedrockAgent
Amazon Bedrock Runtime Agents	`arn:aws:states:::aws-sdk:bedrockagentruntime: [apiAction]`	18 de enero de 2024	BedrockAgentRuntime
Amazon Bedrock Runtime	`arn:aws:states:::aws-sdk:bedrockruntime: [apiAction]`	18 de enero de 2024	BedrockRuntime
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanroomsml: [apiAction]`	18 de enero de 2024	CleanRoomsMl
Amazon CloudFront KeyValueStore	`arn:aws:states:::aws-sdk:cloudfrontkeyvaluestore: [apiAction]`	18 de enero de 2024	CloudFrontKeyValueStore
CodeGuru Security	`arn:aws:states:::aws-sdk:codegurusecurity: [apiAction]`	18 de enero de 2024	CodeGuruSecurity
Centro de optimización de costes de AWS	`arn:aws:states:::aws-sdk:costoptimizationhub: [apiAction]`	18 de enero de 2024	CostOptimizationHub
Amazon DataZone	`arn:aws:states:::aws-sdk:datazone: [apiAction]`	18 de enero de 2024	DataZone

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon EKS Auth	`arn:aws:states:::aws-sdk:eksauth: [apiAction]`	18 de enero de 2024	EksAuth
AWS Entity Resolution	`arn:aws:states:::aws-sdk:entityresolution: [apiAction]`	18 de enero de 2024	EntityResolution
capa gratuita de AWS	`arn:aws:states:::aws-sdk:freetier: [apiAction]`	18 de enero de 2024	FreeTier
Amazon Inspector Scan	`arn:aws:states:::aws-sdk:inspectorscan: [apiAction]`	18 de enero de 2024	InspectorScan
AWS Launch Wizard	`arn:aws:states:::aws-sdk:launchwizard: [apiAction]`	18 de enero de 2024	LaunchWizard
Amazon Managed Blockchain Query	`arn:aws:states:::aws-sdk:managedblockchainquery: [apiAction]`	18 de enero de 2024	ManagedBlockchainQuery
AWS Elemental MediaPackage V2	`arn:aws:states:::aws-sdk:mediapackagev2: [apiAction]`	18 de enero de 2024	MediaPackageV2
AWS HealthImaging	`arn:aws:states:::aws-sdk:medicalimaging: [apiAction]`	18 de enero de 2024	MedicalImaging

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Network Manager	`arn:aws:states:::aws-sdk:networkmanager: [apiAction]`	18 de enero de 2024	NetworkManager
AWS Payment Cryptography	`arn:aws:states:::aws-sdk:paymentcryptography: [apiAction]`	18 de enero de 2024	PaymentCryptography
AWS Payment Cryptography Data	`arn:aws:states:::aws-sdk:paymentcryptographydata: [apiAction]`	18 de enero de 2024	PaymentCryptographyData
AWS Private CA Connector for Active Directory	`arn:aws:states:::aws-sdk:pcaconnectorad: [apiAction]`	18 de enero de 2024	PcaConnectorAd
Amazon Q Business	`arn:aws:states:::aws-sdk:qbusiness: [apiAction]`	18 de enero de 2024	QBusiness
Amazon Q Connect	`arn:aws:states:::aws-sdk:qconnect: [apiAction]`	18 de enero de 2024	QConnect
AWS re:Post	`arn:aws:states:::aws-sdk:repostspace: [apiAction]`	18 de enero de 2024	Repostspace
Amazon Timestream Query	`arn:aws:states:::aws-sdk:timestreamquery: [apiAction]`	18 de enero de 2024	TimestreamQuery

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Timestream Write	`arn:aws:states:::aws-sdk:timestreamwrite: [apiAction]`	18 de enero de 2024	TimestreamWrite
Trusted Advisor	`arn:aws:states:::aws-sdk:trustedadvisor: [apiAction]`	18 de enero de 2024	TrustedAdvisor
Verified Permissions	`arn:aws:states:::aws-sdk:verifiedpermissions: [apiAction]`	18 de enero de 2024	VerifiedPermissions
Amazon WorkSpaces Thin Client	`arn:aws:states:::aws-sdk:workspacesthinclient: [apiAction]`	18 de enero de 2024	WorkSpacesThinClient
AWS CloudTrail Data	`arn:aws:states:::aws-sdk:cloudtraildata: [apiAction]`	16 de junio de 2023	CloudTrailData
Amazon CloudWatch Internet Monitor	`arn:aws:states:::aws-sdk:internetmonitor: [apiAction]`	16 de junio de 2023	InternetMonitor
Amazon Interactive Video Service RealTime	`arn:aws:states:::aws-sdk:ivsrealtime: [apiAction]`	16 de junio de 2023	IvsRealTime
AWS IoT TwinMaker	`arn:aws:states:::aws-sdk:iottwinmaker: [apiAction]`	16 de junio de 2023	IoTTwinMaker

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon OpenSearch Ingestion	`arn:aws:states:::aws-sdk:osis: [apiAction]`	16 de junio de 2023	Osis
AWS Telco Network Builder	`arn:aws:states:::aws-sdk: tnb:[apiAction]`	16 de junio de 2023	Tnb
Amazon VPC Lattice	`arn:aws:states:::aws-sdk:vpclattice: [apiAction]`	16 de junio de 2023	VpcLattice
Amazon Chime Media Pipelines	`arn:aws:states:::aws-sdk:chimesdkmediapipelines: [apiAction]`	17 de febrero de 2023	ChimeSdkMediaPipelines
Amazon Chime Voice	`arn:aws:states:::aws-sdk:chimesdkvoice: [apiAction]`	17 de febrero de 2023	ChimeSdkVoice
Amazon CodeCatalyst	`arn:aws:states:::aws-sdk:codecatalyst: [apiAction]`	17 de febrero de 2023	CodeCatalyst
Amazon Connect Cases	`arn:aws:states:::aws-sdk:connectcases: [apiAction]`	17 de febrero de 2023	ConnectCases
Amazon DocumentDB Elastic Clusters	`arn:aws:states:::aws-sdk:docdbelastic: [apiAction]`	17 de febrero de 2023	DocDbElastic

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon EMR Serverless	`arn:aws:states:::aws-sdk:emrserverless: [apiAction]`	17 de febrero de 2023	EmrServerless
Amazon IVS Chat	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	17 de febrero de 2023	Ivs
Amazon Kendra Intelligent Ranking	`arn:aws:states:::aws-sdk:kendraranking: [apiAction]`	17 de febrero de 2023	KendraRanking
AWS HealthOmics	`arn:aws:states:::aws-sdk:omics: [apiAction]`	17 de febrero de 2023	Omics
Amazon Redshift Serverless	`arn:aws:states:::aws-sdk:redshiftserverless: [apiAction]`	17 de febrero de 2023	RedshiftServerless
Amazon Security Lake	`arn:aws:states:::aws-sdk:securitylake: [apiAction]`	17 de febrero de 2023	SecurityLake
AWS Backup Storage	`arn:aws:states:::aws-sdk:backupstorage: [apiAction]`	17 de febrero de 2023	BackupStorage
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanrooms: [apiAction]`	17 de febrero de 2023	CleanRooms
AWS Control Tower	`arn:aws:states:::aws-sdk:controltower: [apiAction]`	17 de febrero de 2023	ControlTower

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Health	`arn:aws:states:::aws-sdk:health: [apiAction]`	17 de febrero de 2023	Health
AWS IoT FleetWise	`arn:aws:states:::aws-sdk:iotfleetwise: [apiAction]`	17 de febrero de 2023	IotFleetWise
AWS Mainframe Modernization	`arn:aws:states:::aws-sdk:m2: [apiAction]`	17 de febrero de 2023	M2
Orquestador de AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhuborchestrator: [apiAction]`	17 de febrero de 2023	Migration HubOrchestrator
AWS Private 5G	`arn:aws:states:::aws-sdk:privatenetworks: [apiAction]`	17 de febrero de 2023	PrivateNetworks
Explorador de recursos de AWS	`arn:aws:states:::aws-sdk:resourceexplorer2: [apiAction]`	17 de febrero de 2023	ResourceExplorer2
AWS SimSpace Weaver	`arn:aws:states:::aws-sdk:simspaceweaver: [apiAction]`	17 de febrero de 2023	SimSpaceWeaver
AWS Support App	`arn:aws:states:::aws-sdk:supportapp: [apiAction]`	17 de febrero de 2023	SupportApp

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
CloudWatch Observability Access Manager	`arn:aws:states:::aws-sdk:oam:` *`[apiAction]`*	17 de febrero de 2023	Oam
EventBridge Pipes	`arn:aws:states:::aws-sdk:pipes:` *`[apiAction]`*	17 de febrero de 2023	Pipes
EventBridge Scheduler	`arn:aws:states:::aws-sdk:scheduler:` *`[apiAction]`*	17 de febrero de 2023	Scheduler
IAM Roles Anywhere	`arn:aws:states:::aws-sdk:rolesanywhere:` *`[apiAction]`*	17 de febrero de 2023	RolesAnywhere
Kinesis Video WebRTC Storage	`arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage:` *`[apiAction]`*	17 de febrero de 2023	KinesisVideoWebRtcStorage
License Manager Linux Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions:` *`[apiAction]`*	17 de febrero de 2023	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerusersubscriptions:` *`[apiAction]`*	17 de febrero de 2023	LicenseManagerUserSubscriptions
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless:` *`[apiAction]`*	17 de febrero de 2023	OpenSearchServerless

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift: [apiAction]`	17 de febrero de 2023	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial: [apiAction]`	17 de febrero de 2023	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics: [apiAction]`	17 de febrero de 2023	SageMakerMetrics
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap: [apiAction]`	17 de febrero de 2023	SsmSap
AWS Account Management	`arn:aws:states:::aws-sdk:account: [apiAction]`	19 de abril de 2022	Account
AWS Amplify	`arn:aws:states:::aws-sdk:amplify: [apiAction]`	30 de septiembre de 2021	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh: [apiAction]`	30 de septiembre de 2021	AppMesh
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner: [apiAction]`	30 de septiembre de 2021	AppRunner

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig: [apiAction]`	30 de septiembre de 2021	AppConfig
AWS AppConfig Data	`arn:aws:states:::aws-sdk:appconfigdata: [apiAction]`	19 de abril de 2022	AppConfigData
AWS AppSync	`arn:aws:states:::aws-sdk:appsync: [apiAction]`	30 de septiembre de 2021	AppSync
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery: [apiAction]` ***	30 de septiembre de 2021	ApplicationDiscovery
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn: [apiAction]`	30 de septiembre de 2021	Mgn
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager: [apiAction]`	30 de septiembre de 2021	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans: [apiAction]`	30 de septiembre de 2021	AutoScalingPlans
AWS Backup	`arn:aws:states:::aws-sdk:backup: [apiAction]`	30 de septiembre de 2021	Backup

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Backup gateway	`arn:aws:states:::aws-sdk:backupgateway: [apiAction]`	19 de abril de 2022	BackupGateway
AWS Batch	`arn:aws:states:::aws-sdk:batch: [apiAction]`	30 de septiembre de 2021	Batch
AWS Billing Conductor	`arn:aws:states:::aws-sdk:billingconductor: [apiAction]`	26 de julio de 2022	Billingconductor
AWS Budgets	`arn:aws:states:::aws-sdk:budgets: [apiAction]`	30 de septiembre de 2021	Budgets
AWS Certificate Manager	`arn:aws:states:::aws-sdk:acm: [apiAction]`	30 de septiembre de 2021	Acm
AWS Private Certificate Authority	`arn:aws:states:::aws-sdk:acmpca: [apiAction]`	30 de septiembre de 2021	AcmPca
AWS Cloud Map	`arn:aws:states:::aws-sdk:servicediscovery: [apiAction]`	30 de septiembre de 2021	ServiceDiscovery
AWS Cloud9	`arn:aws:states:::aws-sdk:cloud9: [apiAction]`	30 de septiembre de 2021	Cloud9
AWS CloudFormation	`arn:aws:states:::aws-sdk:cloudformation: [apiAction]`	30 de septiembre de 2021	CloudFormation

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsm:` *[apiAction]*	30 de septiembre de 2021	CloudHsm
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsmv2:` *[apiAction]*	30 de septiembre de 2021	CloudHsmV2
AWS CloudTrail	`arn:aws:states:::aws-sdk:cloudtrail:` *[apiAction]*	30 de septiembre de 2021	CloudTrail
AWS Cloud Control	`arn:aws:states:::aws-sdk:cloudcontrol:` *[apiAction]*	19 de abril de 2022	CloudControl
AWS CodeBuild	`arn:aws:states:::aws-sdk:codebuild:` *[apiAction]*	30 de septiembre de 2021	CodeBuild
AWS CodeCommit	`arn:aws:states:::aws-sdk:codecommit:` *[apiAction]*	30 de septiembre de 2021	CodeCommit
AWS CodeDeploy	`arn:aws:states:::aws-sdk:codedeploy:` *[apiAction]* *** —	30 de septiembre de 2021	CodeDeploy
AWS CodePipeline	`arn:aws:states:::aws-sdk:codepipeline:` *[apiAction]*	30 de septiembre de 2021	CodePipeline
AWS CodeStar	`arn:aws:states:::aws-sdk:codestar:` *[apiAction]*	30 de septiembre de 2021	CodeStar

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications:` *[apiAction]*	30 de septiembre de 2021	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections:` *[apiAction]*	30 de septiembre de 2021	CodeStarConnections
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer:` *[apiAction]*	30 de septiembre de 2021	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config:` *[apiAction]*	30 de septiembre de 2021	Config
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer:` *[apiAction]*	30 de septiembre de 2021	CostExplorer
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport:` *[apiAction]*	30 de septiembre de 2021	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange:` *[apiAction]*	30 de septiembre de 2021	DataExchange
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline:` *[apiAction]*	30 de septiembre de 2021	DataPipeline

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS DataSync	`arn:aws:states:::aws-sdk:datasync:` *[apiAction]*	30 de septiembre de 2021	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration:` *[apiAction]*	30 de septiembre de 2021	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm:` *[apiAction]*	30 de septiembre de 2021	DeviceFarm
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect:` *[apiAction]* *** —	30 de septiembre de 2021	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory:` *[apiAction]*	30 de septiembre de 2021	Directory
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect:` *[apiAction]*	30 de septiembre de 2021	Ec2InstanceConnect
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk:` *[apiAction]*	30 de septiembre de 2021	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive:` *[apiAction]*	30 de septiembre de 2021	MediaLive
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage:` *[apiAction]* *** —	30 de septiembre de 2021	MediaPackage

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod: [apiAction]`	30 de septiembre de 2021	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore: [apiAction]`	30 de septiembre de 2021	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis: [apiAction]`	30 de septiembre de 2021	Fis
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms: [apiAction]`	30 de septiembre de 2021	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue: [apiAction]`	30 de septiembre de 2021	Glue
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew: [apiAction]`	30 de septiembre de 2021	DataBrew
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass: [apiAction]`	30 de septiembre de 2021	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation: [apiAction]`	30 de septiembre de 2021	GroundStation
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam: [apiAction]`	30 de septiembre de 2021	Iam

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS IoT	`arn:aws:states:::aws-sdk:iot:` *[apiAction]* ***	30 de septiembre de 2021	Iot
AWS IoT 1-Click	`arn:aws:states:::aws-sdk:iot1clickprojects:` *[apiAction]*	30 de septiembre de 2021	Iot1ClickProjects
AWS IoT Analytics	`arn:aws:states:::aws-sdk:iotanalytics:` *[apiAction]*	30 de septiembre de 2021	IoTAnalytics
AWS IoT Core Device Advisor	`arn:aws:states:::aws-sdk:iotdeviceadvisor:` *[apiAction]* ***	30 de septiembre de 2021	IotDeviceAdvisor
AWS IoT Events	`arn:aws:states:::aws-sdk:iotevents:` *[apiAction]*	30 de septiembre de 2021	IotEvents
Datos de AWS IoT Events	`arn:aws:states:::aws-sdk:ioteventsdata:` *[apiAction]*	30 de septiembre de 2021	IotEventsData
AWS IoT Fleet Hub	`arn:aws:states:::aws-sdk:iotfleethub:` *[apiAction]*	30 de septiembre de 2021	IoTFleetHub
AWS IoT Greengrass Version 2	`arn:aws:states:::aws-sdk:greengrassv2:` *[apiAction]*	30 de septiembre de 2021	GreengrassV2
Datos de trabajos de AWS IoT Plane	`arn:aws:states:::aws-sdk:iotjobsdataplane:` *[apiAction]*	30 de septiembre de 2021	IotJobsDataPlane

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS IoT Secure Tunneling	`arn:aws:states:::aws-sdk:iotsecuretunneling: [apiAction]`	30 de septiembre de 2021	IoTSecure Tunneling
AWS IoT SiteWise	`arn:aws:states:::aws-sdk:iotsitewise: [apiAction]`	30 de septiembre de 2021	IoTSiteWise
AWS IoT Wireless	`arn:aws:states:::aws-sdk:iotwireless: [apiAction]`	30 de septiembre de 2021	IotWireless
AWS Key Management Service	`arn:aws:states:::aws-sdk:kms: [apiAction]`	30 de septiembre de 2021	Kms
AWS Lake Formation	`arn:aws:states:::aws-sdk:lakeformation: [apiAction]`	30 de septiembre de 2021	LakeFormation
AWS Lambda	`arn:aws:states:::aws-sdk:lambda: [apiAction]` ***	30 de septiembre de 2021	Lambda
AWS License Manager	`arn:aws:states:::aws-sdk:licensemanager: [apiAction]`	30 de septiembre de 2021	LicenseManager
AWS Marketplace	`arn:aws:states:::aws-sdk:marketplacecatalog: [apiAction]`	30 de septiembre de 2021	MarketplaceCatalog

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Marketplace Commerce Analytics	`arn:aws:states:::aws-sdk:marketplacecommerceanalytics:` *[apiAction]*	30 de septiembre de 2021	MarketplaceCommerceAnalytics
AWS Marketplace Entitlement Service	`arn:aws:states:::aws-sdk:marketplaceentitlement:` *[apiAction]*	30 de septiembre de 2021	MarketplaceEntitlement
AWS Elemental MediaTailor	`arn:aws:states:::aws-sdk:mediatailor:` *[apiAction]*	30 de septiembre de 2021	MediaTailor
AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhub:` *[apiAction]*	30 de septiembre de 2021	MigrationHub
AWS Migration Hub Config	`arn:aws:states:::aws-sdk:migrationhubconfig:` *[apiAction]*	30 de septiembre de 2021	MigrationHubConfig
Recomendaciones de estrategias de AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhubstrategy:` *[apiAction]*	19 de abril de 2022	MigrationHubStrategy
AWS Mobile	`arn:aws:states:::aws-sdk:mobile:` *[apiAction]*	30 de septiembre de 2021	
AWS Network Firewall	`arn:aws:states:::aws-sdk:networkfirewall:` *[apiAction]*	30 de septiembre de 2021	NetworkFirewall

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS OpsWorks	`arn:aws:states:::aws-sdk:opsworks:` *[apiAction]*	30 de septiembre de 2021	OpsWorks
AWS OpsWorks CM	`arn:aws:states:::aws-sdk:opsworkscm:` *[apiAction]*	30 de septiembre de 2021	OpsWorksCm
AWS Organizations	`arn:aws:states:::aws-sdk:organizations:` *[apiAction]*	30 de septiembre de 2021	Organizations
AWS Outposts	`arn:aws:states:::aws-sdk:outposts:` *[apiAction]*	30 de septiembre de 2021	Outposts
AWS Panorama	`arn:aws:states:::aws-sdk:panorama:` *[apiAction]*	19 de abril de 2022	Panorama
Amazon Relational Database Service Performance Insights	`arn:aws:states:::aws-sdk:pi:` *[apiAction]*	30 de septiembre de 2021	Pi
Lista de precios de AWS	`arn:aws:states:::aws-sdk:pricing:` *[apiAction]*	30 de septiembre de 2021	Pricing
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rdsdata:` *[apiAction]* ***	30 de septiembre de 2021	RdsData

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Resilience Hub	`arn:aws:states:::aws-sdk:resiliencehub: [apiAction]`	19 de abril de 2022	Resiliencehub
AWS Resource Access Manager	`arn:aws:states:::aws-sdk:ram: [apiAction]`	30 de septiembre de 2021	Ram
AWS Resource Groups	`arn:aws:states:::aws-sdk:resourcegroups: [apiAction]`	30 de septiembre de 2021	ResourceGroups
AWS Resource Groups Tagging API	`arn:aws:states:::aws-sdk:resourcegroupstaggingapi: [apiAction]`	30 de septiembre de 2021	ResourceGroupsTaggingApi
AWS RoboMaker	`arn:aws:states:::aws-sdk:robomaker: [apiAction]`	30 de septiembre de 2021	RoboMaker
AWS IAM Identity Center	`arn:aws:states:::aws-sdk:identitystore: [apiAction]`	30 de septiembre de 2021	Identitystore
IAM Identity Center OIDC	`arn:aws:states:::aws-sdk:ssooidc: [apiAction]`	30 de septiembre de 2021	SsoOidc
AWS Secrets Manager	`arn:aws:states:::aws-sdk:secretsmanager: [apiAction]`	30 de septiembre de 2021	SecretsManager

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Security Token Service	`arn:aws:states:::aws-sdk:sts:` *`[apiAction]`* ***	30 de septiembre de 2021	Sts
AWS Security Hub	`arn:aws:states:::aws-sdk:securityhub:` *`[apiAction]`*	30 de septiembre de 2021	SecurityHub
AWS Server Migration Service	`arn:aws:states:::aws-sdk:sms:` *`[apiAction]`*	30 de septiembre de 2021	Sms
AWS Service Catalog	`arn:aws:states:::aws-sdk:servicecatalog:` *`[apiAction]`*	30 de septiembre de 2021	ServiceCatalog
AWS Service Catalog AppRegistry	`arn:aws:states:::aws-sdk:servicecatalogappregistry:` *`[apiAction]`*	30 de septiembre de 2021	ServiceCatalogAppRegistry
AWS Shield	`arn:aws:states:::aws-sdk:shield:` *`[apiAction]`* ***	30 de septiembre de 2021	Shield
AWS Signer	`arn:aws:states:::aws-sdk:signer:` *`[apiAction]`*	30 de septiembre de 2021	Signer
IAM Identity Center	`arn:aws:states:::aws-sdk:sso:` *`[apiAction]`*	30 de septiembre de 2021	Sso

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
IAM Identity Center Admin	`arn:aws:states:::aws-sdk:ssoadmin:` *[apiAction]*	30 de septiembre de 2021	SsoAdmin
AWS Step Functions	`arn:aws:states:::aws-sdk:sfn:` *[apiAction]*	30 de septiembre de 2021	Sfn
AWS Storage Gateway	`arn:aws:states:::aws-sdk:storagegateway:` *[apiAction]*	30 de septiembre de 2021	StorageGateway
AWS Support	`arn:aws:states:::aws-sdk:support:` *[apiAction]*	30 de septiembre de 2021	Support
AWS Systems Manager Incident Manager	`arn:aws:states:::aws-sdk:ssmincidents:` *[apiAction]*		SsmIncidents
AWS Transfer Family	`arn:aws:states:::aws-sdk:transfer:` *[apiAction]*	30 de septiembre de 2021	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf:` *[apiAction]*	30 de septiembre de 2021	Waf
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional:` *[apiAction]*	30 de septiembre de 2021	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2:` *[apiAction]*	30 de septiembre de 2021	Wafv2

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	30 de septiembre de 2021	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	30 de septiembre de 2021	XRay
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering: [apiAction]`	30 de septiembre de 2021	MarketplaceMetering
AWS Serverless Application Repository	`arn:aws:states:::aws-sdk:serverlessapplicationrepository: [apiAction]`	30 de septiembre de 2021	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	`arn:aws:states:::aws-sdk:accessanalyzer: [apiAction]`	30 de septiembre de 2021	AccessAnalyzer
Alexa for Business	`arn:aws:states:::aws-sdk:alexaforbusiness: [apiAction]`	30 de septiembre de 2021	AlexaForBusiness
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigateway: [apiAction]`	30 de septiembre de 2021	ApiGateway
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigatewayv2: [apiAction]`	30 de septiembre de 2021	ApiGatewayV2

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon AppIntegrations	`arn:aws:states:::aws-sdk:appintegrations: [apiAction]`	30 de septiembre de 2021	AppIntegrations
Amazon AppStream 2.0	`arn:aws:states:::aws-sdk:appstream: [apiAction]`	30 de septiembre de 2021	AppStream
Amazon AppFlow	`arn:aws:states:::aws-sdk:appflow: [apiAction]`	30 de septiembre de 2021	Appflow
Amazon Athena	`arn:aws:states:::aws-sdk:athena: [apiAction]`	30 de septiembre de 2021	Athena
Amazon Augmented AI	`arn:aws:states:::aws-sdk:sagemakera2iruntime: [apiAction]`	30 de septiembre de 2021	SageMaker A2IRuntime
Amazon Braket	`arn:aws:states:::aws-sdk:braket: [apiAction]`	30 de septiembre de 2021	Braket
Amazon Chime	`arn:aws:states:::aws-sdk:chime: [apiAction]`	30 de septiembre de 2021	Chime
Amazon Chime Meetings	`arn:aws:states:::aws-sdk:chimesdkmeetings: [apiAction]`	19 de abril de 2022	ChimeSdkMeetings

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Cloud Directory	`arn:aws:states:::aws-sdk:clouddirectory: [apiAction]`	30 de septiembre de 2021	CloudDirectory
Amazon CloudFront	`arn:aws:states:::aws-sdk:cloudfront: [apiAction]`	30 de septiembre de 2021	CloudFront
Amazon CloudSearch	`arn:aws:states:::aws-sdk:cloudsearch: [apiAction]`	30 de septiembre de 2021	CloudSearch
Amazon CloudWatch	`arn:aws:states:::aws-sdk:cloudwatch: [apiAction]`	30 de septiembre de 2021	CloudWatch
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights: [apiAction]`	30 de septiembre de 2021	ApplicationInsights
CloudWatch Evidently	`arn:aws:states:::aws-sdk:evidently: [apiAction]`	19 de abril de 2022	Evidently
Amazon CloudWatch Logs	`arn:aws:states:::aws-sdk:cloudwatchlogs: [apiAction]`	30 de septiembre de 2021	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum: [apiAction]`	19 de abril de 2022	Rum

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics: [apiAction]`	30 de septiembre de 2021	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler: [apiAction]`	30 de septiembre de 2021	CodeGuruProfiler
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer: [apiAction]`	30 de septiembre de 2021	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity: [apiAction]`	30 de septiembre de 2021	CognitoIdentity
Amazon Cognito Identity Provider	`arn:aws:states:::aws-sdk:cognitoidentityprovider: [apiAction]`	30 de septiembre de 2021	CognitoIdentityProvider
Amazon Cognito Sync	`arn:aws:states:::aws-sdk:cognitosync: [apiAction]`	30 de septiembre de 2021	CognitoSync
Amazon Comprehend	`arn:aws:states:::aws-sdk:comprehend: [apiAction]`	30 de septiembre de 2021	Comprehend
Amazon Comprehend Medical	`arn:aws:states:::aws-sdk:comprehendmedical: [apiAction]` *** —	30 de septiembre de 2021	ComprehendMedical

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Connect Contact Lens	`arn:aws:states:::aws-sdk:connectcontactlens: [apiAction]`	30 de septiembre de 2021	ConnectContactLens
Amazon Connect Participant Service	`arn:aws:states:::aws-sdk:connectparticipant: [apiAction]`	30 de septiembre de 2021	ConnectParticipant
Amazon Connect	`arn:aws:states:::aws-sdk:connect: [apiAction]`	30 de septiembre de 2021	Connect
Amazon Connect Voice ID	`arn:aws:states:::aws-sdk:voiceid: [apiAction]`	19 de abril de 2022	VoiceId
Amazon Connect Wisdom	`arn:aws:states:::aws-sdk:wisdom: [apiAction]`	19 de abril de 2022	Wisdom
Amazon Data Lifecycle Manager	`arn:aws:states:::aws-sdk:dlm: [apiAction]`	30 de septiembre de 2021	Dlm
Amazon Detective	`arn:aws:states:::aws-sdk:detective: [apiAction]`	30 de septiembre de 2021	Detective
Amazon DevOps Guru	`arn:aws:states:::aws-sdk:devopsguru: [apiAction]`	30 de septiembre de 2021	DevOpsGuru

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon DocumentDB (with MongoDB compatibility)	`arn:aws:states:::aws-sdk:docdb: [apiAction]`	30 de septiembre de 2021	DocDb
Amazon DynamoDB	`arn:aws:states:::aws-sdk:dynamodb: [apiAction]`	30 de septiembre de 2021	DynamoDb
Amazon DynamoDB Streams	`arn:aws:states:::aws-sdk:dynamodbstreams: [apiAction]`	30 de septiembre de 2021	DynamoDbStreams
Amazon EC2 Container Registry	`arn:aws:states:::aws-sdk:ecr: [apiAction]`	30 de septiembre de 2021	Ecr
Amazon EC2 Container Service	`arn:aws:states:::aws-sdk:ecs: [apiAction]`	30 de septiembre de 2021	Ecs
Amazon EC2 Systems Manager	`arn:aws:states:::aws-sdk:ssm: [apiAction]`	30 de septiembre de 2021	Ssm
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers: [apiAction]` *** —	30 de septiembre de 2021	EmrContainers
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache: [apiAction]`	30 de septiembre de 2021	ElastiCache

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference: [apiAction]`	30 de septiembre de 2021	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs: [apiAction]`	30 de septiembre de 2021	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2: [apiAction]`	30 de septiembre de 2021	Ec2
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic: [apiAction]`	30 de septiembre de 2021	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs: [apiAction]` *** —	30 de septiembre de 2021	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks: [apiAction]`	30 de septiembre de 2021	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr: [apiAction]`	30 de septiembre de 2021	Emr
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder: [apiAction]` *** —	30 de septiembre de 2021	ElasticTranscoder
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch: [apiAction]`	30 de septiembre de 2021	Elasticsearch

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:opensearch:` *`[apiAction]`*	19 de abril de 2022	OpenSearch
Amazon EventBridge	`arn:aws:states:::aws-sdk:eventbridge:` *`[apiAction]`*	30 de septiembre de 2021	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx:` *`[apiAction]`*	30 de septiembre de 2021	FSx
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery:` *`[apiAction]`*	30 de septiembre de 2021	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast:` *`[apiAction]`*	30 de septiembre de 2021	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector:` *`[apiAction]`*	30 de septiembre de 2021	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift:` *`[apiAction]`*	30 de septiembre de 2021	Amazon GameLift
Amazon GameSparks	`arn:aws:states:::aws-sdk:gamesparks:` *`[apiAction]`*	27 de julio de 2022	GameSparks
Amazon S3 Glacier	`arn:aws:states:::aws-sdk:glacier:` *`[apiAction]`*	30 de septiembre de 2021	Glacier

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon GuardDuty	`arn:aws:states:::aws-sdk:guardduty:` *`[apiAction]`*	30 de septiembre de 2021	GuardDuty
AWS HealthLake	`arn:aws:states:::aws-sdk:healthlake:` *`[apiAction]`*	30 de septiembre de 2021	HealthLake
Amazon Honeycode	`arn:aws:states:::aws-sdk:honeycode:` *`[apiAction]`*	30 de septiembre de 2021	Honeycode
Amazon Inspector	`arn:aws:states:::aws-sdk:inspector:` *`[apiAction]`*	30 de septiembre de 2021	Inspector
Amazon Inspector V2	`arn:aws:states:::aws-sdk:inspector2:` *`[apiAction]`*	19 de abril de 2022	Inspector2
Amazon Interactive Video Service	`arn:aws:states:::aws-sdk:ivs:` *`[apiAction]`*	30 de septiembre de 2021	Ivs
Amazon Kendra	`arn:aws:states:::aws-sdk:kendra:` *`[apiAction]`*	30 de septiembre de 2021	Kendra
Amazon Kinesis	`arn:aws:states:::aws-sdk:kinesis:` *`[apiAction]`* *** —	30 de septiembre de 2021	Kinesis
Amazon Kinesis Analytics	`arn:aws:states:::aws-sdk:kinesisanalytics:` *`[apiAction]`*	30 de septiembre de 2021	KinesisAnalytics

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Kinesis Analytics V2	`arn:aws:states:::aws-sdk:kinesisanalyticsv2: [apiAction]`	30 de septiembre de 2021	KinesisAnalyticsV2
Amazon Kinesis Firehose	`arn:aws:states:::aws-sdk:firehose: [apiAction]`	30 de septiembre de 2021	Firehose
Amazon Kinesis Video Signaling Channels	`arn:aws:states:::aws-sdk:kinesisvideosignaling: [apiAction]`	30 de septiembre de 2021	KinesisVideoSignaling
Amazon Kinesis Video Streams	`arn:aws:states:::aws-sdk:kinesisvideo: [apiAction]`	30 de septiembre de 2021	KinesisVideo
Amazon Kinesis Video Streams Archived Media	`arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia: [apiAction]`	30 de septiembre de 2021	KinesisVideoArchivedMedia
Amazon Kinesis video stream	`arn:aws:states:::aws-sdk:kinesisvideomedia: [apiAction]`	30 de septiembre de 2021	KinesisVideoMedia
Amazon Lex Model Building Service	`arn:aws:states:::aws-sdk:lexmodelbuilding: [apiAction]`	30 de septiembre de 2021	LexModelBuilding
Amazon Lex Model Building Service V2	`arn:aws:states:::aws-sdk:lexmodelsv2: [apiAction]`	30 de septiembre de 2021	LexModelsV2

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Lex	`arn:aws:states:::aws-sdk:lexruntime:` *[apiAction]*	30 de septiembre de 2021	LexRuntime
Amazon Lex Runtime V2	`arn:aws:states:::aws-sdk:lexruntimev2:` *[apiAction]* ***	30 de septiembre de 2021	LexRuntimeV2
Amazon Lightsail	`arn:aws:states:::aws-sdk:lightsail:` *[apiAction]*	30 de septiembre de 2021	Lightsail
Amazon Location Service	`arn:aws:states:::aws-sdk:location:` *[apiAction]*	30 de septiembre de 2021	Location
Amazon Lookout for Equipment	`arn:aws::states:::aws-sdk:lookoutequipment:` *[apiAction]*	30 de septiembre de 2021	LookoutEquipment
Amazon Lookout for Metrics	`arn:aws:states:::aws-sdk:lookoutmetrics:` *[apiAction]*	30 de septiembre de 2021	LookoutMetrics
Amazon Lookout for Vision	`arn:aws:states:::aws-sdk:lookoutvision:` *[apiAction]*	30 de septiembre de 2021	LookoutVision
Amazon MQ	`arn:aws:states:::aws-sdk:mq:` *[apiAction]*	30 de septiembre de 2021	Mq
Amazon Macie	`arn:aws:states:::aws-sdk:macie:` *[apiAction]*	30 de septiembre de 2021	

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Macie 2	`arn:aws:states:::aws-sdk:macie2: [apiAction]`	30 de septiembre de 2021	Macie2
Amazon Managed Blockchain	`arn:aws:states:::aws-sdk:managedblockchain: [apiAction]`	30 de septiembre de 2021	ManagedBlockchain
Amazon Managed Grafana	`arn:aws:states:::aws-sdk:grafana: [apiAction]`	19 de abril de 2022	Grafana
Amazon Managed Service for Prometheus	`arn:aws:states:::aws-sdk:amp: [apiAction]`	30 de septiembre de 2021	Amp
Amazon Managed Streaming for Apache Kafka	`arn:aws:states:::aws-sdk:kafka: [apiAction]`	30 de septiembre de 2021	Kafka
Amazon MSK Connect	`arn:aws:states:::aws-sdk:kafkaconnect: [apiAction]`	19 de abril de 2022	KafkaConnect
Amazon Managed Workflows for Apache Airflow	`arn:aws:states:::aws-sdk:mwaa: [apiAction]`	30 de septiembre de 2021	Mwaa
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk: [apiAction]`	30 de septiembre de 2021	MTurk
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb: [apiAction]`	19 de abril de 2022	MemoryDB

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble: [apiAction]`	30 de septiembre de 2021	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize: [apiAction]`	30 de septiembre de 2021	Personalize
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents: [apiAction]`	30 de septiembre de 2021	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime: [apiAction]`	30 de septiembre de 2021	PersonalizeRuntime
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint: [apiAction]`	30 de septiembre de 2021	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail: [apiAction]`	30 de septiembre de 2021	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice: [apiAction]`	30 de septiembre de 2021	PinpointSmsVoice
Amazon Pinpoint SMS and Voice V2 Service	`arn:aws:states:::aws-sdk:pinpointsmsvoicev2: [apiAction]`	27 de julio de 2022	PinpointSmsVoiceV2
Amazon Polly	`arn:aws:states:::aws-sdk:polly: [apiAction]`	30 de septiembre de 2021	Polly

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon QLDB	`arn:aws:states:::aws-sdk:qldb: [apiAction]`	30 de septiembre de 2021	Qldb
Amazon QLDB Session	`arn:aws:states:::aws-sdk:qldbsession: [apiAction]`	30 de septiembre de 2021	QldbSession
Amazon QuickSight	`arn:aws:states:::aws-sdk:quicksight: [apiAction]`	30 de septiembre de 2021	QuickSight
Amazon Redshift	`arn:aws:states:::aws-sdk:redshift: [apiAction]`	30 de septiembre de 2021	Redshift
Amazon Redshift Data API	`arn:aws:states:::aws-sdk:redshiftdata: [apiAction]`	30 de septiembre de 2021	RedshiftData
Amazon Rekognition	`arn:aws:states:::aws-sdk:rekognition: [apiAction]`	30 de septiembre de 2021	Rekognition
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rds: [apiAction]`	30 de septiembre de 2021	Rds
Amazon Route 53	`arn:aws:states:::aws-sdk:route53: [apiAction]`	30 de septiembre de 2021	Route53

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Route 53 Recovery Control Config	`arn:aws:states:::aws-sdk:route53recoverycontrolconfig:` *[apiAction]*	30 de septiembre de 2021	Route53RecoveryControlConfig
Amazon Route 53 Domains	`arn:aws:states:::aws-sdk:route53domains:` *[apiAction]*	30 de septiembre de 2021	Route53Domains
Amazon Route 53 Resolver	`arn:aws:states:::aws-sdk:route53resolver:` *[apiAction]*	30 de septiembre de 2021	Route53Resolver
Amazon S3 on Outposts	`arn:aws:states:::aws-sdk:s3outposts:` *[apiAction]*	30 de septiembre de 2021	S3Outposts
Amazon SageMaker Runtime Feature Store Runtime	`arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime:` *[apiAction]*	30 de septiembre de 2021	SageMakerRuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	`arn:aws:states:::aws-sdk:sagemakerruntime:` *[apiAction]*	30 de septiembre de 2021	SageMakerRuntimeRuntime
Amazon SageMaker	`arn:aws:states:::aws-sdk:sagemaker:` *[apiAction]*	30 de septiembre de 2021	SageMakerRuntime
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge:` *[apiAction]*	30 de septiembre de 2021	SagemakerEdge

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses:` *`[apiAction]`*	30 de septiembre de 2021	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2:` *`[apiAction]`*	30 de septiembre de 2021	SesV2
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *`[apiAction]`*	30 de septiembre de 2021	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *`[apiAction]`*	30 de septiembre de 2021	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *`[apiAction]`* ***	30 de septiembre de 2021	S3
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *`[apiAction]`*	30 de septiembre de 2021	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *`[apiAction]`*	30 de septiembre de 2021	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *`[apiAction]`*	30 de septiembre de 2021	Transcribe
Amazon Translate	`arn:aws:states:::aws-sdk:translate:` *`[apiAction]`*	30 de septiembre de 2021	Translate

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon WorkDocs	`arn:aws:states:::aws-sdk:workdocs:` *[apiAction]*	30 de septiembre de 2021	WorkDocs
Amazon WorkMail	`arn:aws:states:::aws-sdk:workmail:` *[apiAction]*	30 de septiembre de 2021	WorkMail
Amazon WorkMail Message Flow	`arn:aws:states:::aws-sdk:workmailmessageflow:` *[apiAction]*	30 de septiembre de 2021	WorkMailMessageFlow
Amazon WorkSpaces	`arn:aws:states:::aws-sdk:workspaces:` *[apiAction]*	30 de septiembre de 2021	WorkSpaces
Amazon WorkSpaces Web	`arn:aws:states:::aws-sdk:workspacesweb:` *[apiAction]*	19 de abril de 2022	WorkSpacesWeb
Amplify	`arn:aws:states:::aws-sdk:amplifybackend:` *[apiAction]*	30 de septiembre de 2021	AmplifyBackend
Amplify UI Builder	`arn:aws:states:::aws-sdk:amplifyuibuilder:` *[apiAction]*	19 de abril de 2022	AmplifyUiBuilder
Application Auto Scaling	`arn:aws:states:::aws-sdk:applicationautoscaling:` *[apiAction]*	30 de septiembre de 2021	ApplicationAutoScaling

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon EC2 Auto Scaling	`arn:aws:states:::aws-sdk:autoscaling: [apiAction]`	30 de septiembre de 2021	Auto Scaling
CodeArtifact	`arn:aws:states:::aws-sdk:codeartifact: [apiAction]`	30 de septiembre de 2021	Codeartifact
DynamoDB Accelerator	`arn:aws:states:::aws-sdk:dax: [apiAction]`	30 de septiembre de 2021	Dax
EC2 Image Builder	`arn:aws:states:::aws-sdk:imagebuilder: [apiAction]`	30 de septiembre de 2021	Imagebuilder
AWS Elastic Disaster Recovery	`arn:aws:states:::aws-sdk:drs: [apiAction]`	19 de abril de 2022	Drs
Elastic Load Balancing	`arn:aws:states:::aws-sdk:elasticloadbalancing: [apiAction]`	30 de septiembre de 2021	ElasticLoadBalancing
Elastic Load Balancing V2	`arn:aws:states:::aws-sdk:elasticloadbalancingv2: [apiAction]`	30 de septiembre de 2021	ElasticLoadBalancingV2
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect: [apiAction]`	30 de septiembre de 2021	MediaConnect

Nombre del servicio	Recurso de estado **Task**	Fecha de admisión	Prefijo de excepción
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control:` *[apiAction]* ***	30 de septiembre de 2021	S3Control
Recycle Bin for Amazon EBS	`arn:aws:states:::aws-sdk:rbin:` *[apiAction]*	19 de abril de 2022	Rbin
Savings Plans	`arn:aws:states:::aws-sdk:savingsplans:` *[apiAction]*	30 de septiembre de 2021	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas:` *[apiAction]*	30 de septiembre de 2021	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas:` *[apiAction]*	30 de septiembre de 2021	ServiceQuotas
AWS Snowball	`arn:aws:states:::aws-sdk:snowball:` *[apiAction]*	30 de septiembre de 2021	Snowball

Acciones de API no admitidas para servicios admitidos

En la siguiente tabla se enumeran las acciones de API no compatibles para las integraciones de servicios AWS del SDK. La columna de la derecha contiene las acciones de API que actualmente no se admiten para el servicio que aparece en la columna de la izquierda.

Nombre del servicio	Acciones de API no admitidas
AWS Application Discovery Service	• `DescribeExportConfigurations`

Nombre del servicio	Acciones de API no admitidas
	• `ExportConfigurations`
Amazon Bedrock	• `InvokeModelWithResponseStream`
Agentes para Amazon Bedrock Runtime	• `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`
AWS IoT Asesor de dispositivos principales	• `ListTestCases`
Amazon Kinesis	• `SubscribeToShard`

Nombre del servicio	Acciones de API no admitidas
AWS Lambda	• `InvokeAsync` • `InvokeWithResponseStream`
Amazon Lex Runtime V2	• `StartConversation`
AWS Elemental MediaPackage	• `RotateChannelCredentials`
Amazon Relational Database Service	• `ExecuteSql`
Amazon Simple Storage Service	• `SelectObjectContent`
Amazon S3 Control	• `SelectObjectContent`
AWS Shield	• `DeleteSubscription`
AWS Security Token Service	• `AssumeRole` • `AssumeRoleWithSAML` • `AssumeRoleWithWebIdentity`

Integraciones de servicios AWS SDK obsoletas

Las siguientes integraciones de servicios AWS del SDK están ahora en desuso:

- AWS Mobile
- Amazon Macie
- AWS IoT RoboRunner

Integraciones optimizadas para Step Functions

Los siguientes temas incluyen las API, los parámetros y la sintaxis de solicitud/respuesta compatibles en Amazon States Language para coordinar otros servicios. AWS Los temas también ofrecen un código de ejemplo. Puede llamar a los servicios de integraciones optimizadas directamente desde Amazon States Language en el campo `Resource` de un estado `Task`.

Puede utilizar tres patrones de integración de servicios:

- Solicitar una respuesta (opción predeterminada): espere una respuesta HTTP y, a continuación, pase al siguiente estado
- Run a Job (`.sync`): espere a que se complete el trabajo
- Espera a que aparezca Callback (`.waitForTaskToken`): pausa un flujo de trabajo hasta que se devuelva un token de tarea

Los flujos de trabajo estándar y los flujos de trabajo exprés admiten las mismas integraciones pero no los mismos patrones de integración.

- La compatibilidad con los patrones de integración es diferente para cada integración.
- Los flujos de trabajo de Express no admiten Run a Job (.sync) ni Wait for Callback (). waitForTaskSímbolo).
- Para obtener más información, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

Standard Workflows

Integraciones de servicios compatibles

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**`.sync`**)	Espere la devolución de la llamada (**`.waitForTaskToken`**)
Integraciones optimizadas	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integraciones de SDK	Más de doscientas	✓		✓

Express Workflows

Integraciones de servicios compatibles

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (`.sync`)	Espere la devolución de la llamada (`.waitForTaskToken`)
Integraciones optimizadas	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	✓		

	Servicio	Respuesta de la solicitud	Ejecutar un trabajo (**.sync**)	Espere la devolución de la llamada (**.waitForTaskToken**)
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS Integraciones de SDK	Más de doscientas	✓		

Llamar a API Gateway con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de API Gateway de la integración del AWS SDK de API Gateway
>
> - `apigateway:invoke:`no tiene equivalente en la integración del servicio del AWS SDK. En su lugar, el servicio optimizado de API Gateway llama directamente a su punto de conexión de API Gateway.

Puede usar Amazon API Gateway para crear, publicar, mantener y supervisar API de REST y HTTP. Para la integración con API Gateway, se debe definir un estado `Task` en Step Functions que llame directamente a un punto de conexión HTTP o REST de API Gateway, sin necesidad de escribir código ni depender de otra infraestructura. Una definición de estado `Task` incluye toda la información necesaria para la llamada a la API. También puede seleccionar distintos métodos de autorización.

Compatibilidad con características de API Gateway

La integración de API Gateway de Step Functions admite algunas características de API Gateway, pero no todas. Para obtener una lista más detallada de las características compatibles, consulte lo siguiente.

- Compatible con las integraciones de la API de REST y HTTP de API Gateway de Step Functions:
  - Autorizadores: IAM (con Signature Version 4), sin autenticación, autorizadores Lambda (basados en parámetros de solicitud y basados en tokens con encabezado personalizado)
  - Tipos de API: regionales
  - Administración de API: nombres de dominio de API Gateway, etapa de API, ruta, parámetros de consulta, cuerpo de la solicitud
- Compatible con la integración de la API de HTTP de API Gateway de Step Functions. No se admite la integración de la API de REST de API Gateway de Step Functions que ofrece la opción de API optimizadas para Edge.
- No compatible con la integración de la API Gateway de Step Functions:
  - Autorizadores: Amazon Cognito, Open ID Connect/OAuth 2.0 nativo, encabezado de autorización para autorizadores Lambda basados en tokens
  - Tipos de API: privadas
  - Administración de API: nombres de dominio personalizados

Para obtener más información acerca de API Gateway y sus API de REST y HTTP, consulte lo siguiente.

- La página Conceptos de Amazon API Gateway.
- Elección entre las API de HTTP y las API de REST en la Guía para desarrolladores de API Gateway.

Formato de solicitudes

Al crear la definición de estado `Task`, Step Functions valida los parámetros, genera la URL necesaria para realizar la llamada y, posteriormente, llama a la API. La respuesta incluye el código de estado HTTP, los encabezados y el cuerpo de respuesta. El formato de solicitud tiene parámetros obligatorios y opcionales.

Parámetros de solicitud obligatorios

- `ApiEndpoint`
  - Tipo: `String`
  - El nombre de host de una URL de API Gateway. El formato es *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`.

    El ID de API solo puede contener una combinación de los siguientes caracteres alfanuméricos: `0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - Tipo: `Enum`
  - El método HTTP, que debe ser uno de los siguientes:
    - `GET`
    - `POST`
    - `PUT`
    - `DELETE`
    - `PATCH`
    - `HEAD`
    - `OPTIONS`

Parámetros de solicitudes opcionales

- `Headers`
  - Tipo: `JSON`
  - Los encabezados HTTP permiten una lista de valores asociados a la misma clave.
- `Stage`
  - Tipo: `String`

  - El nombre de la fase en la que se implementa la API en API Gateway. Es opcional para cualquier API de HTTP que utilice la fase `$default`.
- `Path`
  - Tipo: `String`
  - Parámetros de ruta que se anexan después del punto de conexión de la API.
- `QueryParameters`
  - Tipo: `JSON`
  - Las cadenas de consulta solo permiten una lista de valores asociados a la misma clave.
- `RequestBody`
  - Tipo: `JSON` o `String`
  - El cuerpo de la solicitud HTTP Su tipo puede ser un objeto `JSON` o`String`. `RequestBody` solo es compatible con los métodos HTTP `PATCH`, `POST` y `PUT`.
- `AllowNullValues`
  - Tipo: `BOOLEAN` — valor predeterminado: `false`
  - Con la configuración predeterminada, los valores nulos que se encuentren en el estado de entrada de la solicitud no se enviarán a tu API. En el siguiente ejemplo, el `category` campo no se incluirá en la solicitud, a menos que `AllowNullValues` esté configurado `true` en la definición de la máquina de estados.

```
{
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "category": null
    }
}
```

> **Note**
>
> De forma predeterminada, los campos con valores nulos en el estado de entrada de la solicitud no se enviarán a tu API. Puedes forzar el envío de valores nulos a tu API `AllowNullValues` configurándolos `true` en tu definición de máquina de estados.

- `AuthType`
  - Tipo: `JSON`

- El método de autenticación. El método predeterminado es `NO_AUTH`. Los valores permitidos son:
  - `NO_AUTH`
  - `IAM_ROLE`
  - `RESOURCE_POLICY`

  Para obtener más información, consulte Autenticación y autorización.

> **Note**
>
> Por motivos de seguridad, actualmente no se permiten las siguientes claves de encabezado HTTP:
>
> - Cualquiera que tenga el prefijo `X-Forwarded`, `X-Amz` o `X-Amzn`.
> - `Authorization`
> - `Connection`
> - `Content-md5`
> - `Expect`
> - `Host`
> - `Max-Forwards`
> - `Proxy-Authenticate`
> - `Server`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `Www-Authenticate`

En el siguiente ejemplo de código se muestra cómo invocar API Gateway mediante Step Functions.

```
{
    "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"
    }
}
```

Autenticación y autorización

Puede usar los siguientes métodos de autenticación:

- Sin autorización: llama a la API directamente sin ningún método de autorización.
- Rol de IAM: con este método, Step Functions asume el rol de la máquina de estado, firma la solicitud con Signature Version 4 (SiGv4) y, posteriormente, llama a la API.
- Política de recursos: Step Functions autentica la solicitud y luego llama a la API. Debe adjuntar una política de recursos a la API que especifique lo siguiente:
  1. La máquina de estado que invocará API Gateway.

     > **Important**
     >
     > Debe especificar su máquina de estado para limitar el acceso a ella. Si no lo hace, se concederá acceso a cualquier máquina de estado que autentique su solicitud de API Gateway con la autenticación de la política de recursos en su API.

  2. Esa Step Functions es el servicio que llama a API Gateway: `"Service": "states.amazonaws.com"`.
  3. El recurso al que desea obtener acceso, que incluye:
     - La *`region`*.
     - El *`account-id`* en la región especificada.

- El *api-id*.
- El *stage-name*.
- El *HTTP-VERB* (método).
- El *resource-path-specifier*.

Para ver una política de recursos de ejemplo, consulte las Políticas de IAM para Step Functions y API Gateway.

Para obtener más información acerca del formato de recursos, consulte Formato de Resource de permisos para ejecutar la API en API Gateway en la Guía para desarrolladores de API Gateway.

> **Note**
>
> Las políticas de recursos solo se admiten en la API de REST.

Patrones de integración de servicios

La integración de API Gateway admite dos patrones de integración de servicios:

- Respuesta de la solicitud, que es el patrón de integración predeterminado. Permite que Step Functions avance al siguiente paso justo después de recibir una respuesta HTTP.
- Cómo esperar una devolución de llamada con el token de tarea (`.waitForTaskToken`), que espera a que se devuelva un token de tarea con una carga. Para usar el `.waitForTaskToken` patrón, añade .wait ForTaskToken al final del campo Recurso de la definición de tu tarea, tal y como se muestra en el siguiente ejemplo:

```
{
    "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"
    }
}
```

Formato de salida

Debe proporcionar los siguientes parámetros de salida:

Nombre	Tipo	Descripción
`ResponseBody`	`JSON` o `String`	Cuerpo de la respuesta de la llamada a la API.
`Headers`	`JSON`	Encabezados de respuesta.
`StatusCode`	`Integer`	Código de estado HTTP de la respuesta.
`StatusText`	`String`	Texto de estado de la respuesta.

Respuesta de ejemplo:

```
{
    "ResponseBody": {
        "myBills": []
    },
    "Headers": {
        "key": ["value1", "value2"]
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

Control de errores

Cuando se produce un error, se devuelve `error` y `cause` de la siguiente manera:

- Si el código de estado HTTP está disponible, el error se devolverá en el formato `ApiGateway.<HTTP Status Code>`.
- Si el código de estado HTTP no está disponible, el error se devolverá en el formato `ApiGateway.<Exception>`.

En ambos casos, `cause` se devuelve como una cadena.

En el siguiente ejemplo se muestra una respuesta en la que se ha producido un error:

```
{
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> Un código de estado de 2XX indica que se ha realizado correctamente y no se devolverá ningún error. El resto de códigos de estado o excepciones emitidas generarán un error.

Para obtener más información, consulte:

- Conceptos de Amazon API Gateway en la guía para desarrolladores de API Gateway.
- Políticas de IAM para Amazon API Gateway
- Un proyecto de muestra que ilustra cómo Hacer una llamada a API Gateway

Conceptos de Amazon API Gateway en la guía para desarrolladores de API Gateway.

Llamar a Athena con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

En qué se diferencia la integración optimizada de Athena de la integración del SDK de Athena AWS

- Se admite el patrón de integración Ejecutar un trabajo (.sync).
- No hay optimizaciones para el patrón de integración Respuesta de la solicitud.
- No se admite el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.

La integración del AWS Step Functions servicio con Amazon Athena le permite usar Step Functions para iniciar y detener la ejecución de consultas y obtener los resultados de las consultas. Con Step Functions, puede ejecutar consultas de datos ad hoc o programadas y recuperar resultados dirigidos a sus lagos de datos S3. Athena no requiere un servidor, por lo que no hay que configurar ni administrar ninguna infraestructura y solo pagará por las consultas que ejecute.

Para la integración AWS Step Functions con Amazon Athena, debe utilizar las API de integración de servicios de Athena proporcionadas.

Las API de integración de servicios son las mismas que las API de Athena correspondientes. No todas las API admiten todos los patrones de integración, como se muestra en la tabla siguiente.

API	Respuesta de la solicitud	Ejecutar un trabajo (.sync)
`StartQueryExecution`	✓	✓
`StopQueryExecution`	✓	
`GetQueryExecution`	✓	
`GetQueryResults`	✓	

API de Amazon Athena compatibles:

Note

Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o

recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

- `StartQueryExecution`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `ClientRequestToken`
    - `ExecutionParameters`
    - `QueryExecutionContext`
    - `QueryString`
    - `ResultConfiguration`
    - `WorkGroup`
  - `Response syntax`
- `StopQueryExecution`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `QueryExecutionId`
- `GetQueryExecution`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `QueryExecutionId`
  - `Response syntax`
- `GetQueryResults`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `MaxResults`
    - `NextToken`
    - `QueryExecutionId`
  - `Response syntax`

El ejemplo siguiente incluye un estado Task que comienza un trabajo de Athena.

```
"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"
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte. Políticas de IAM para servicios integrados

Administre AWS Batch con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la AWS Batch integración optimizada de la integración del AWS BatchAWS SDK
>
> - El patrón de integración Ejecutar un trabajo (.sync) está disponible.
>
> Tenga en cuenta que no hay optimizaciones para los patrones de integración Respuesta de la solicitud o Cómo esperar una devolución de llamada con el token de tarea.

AWS Batch APIs compatibles:

- `SubmitJob`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `ArrayProperties`

- ContainerOverrides
- DependsOn
- JobDefinition
- JobName
- JobQueue
- Parameters
- RetryStrategy
- Timeout
- Tags

- Sintaxis de la respuesta

> **Los parámetros de Step Functions se expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

A continuación se incluye un `Task` estado en el que se envía un AWS Batch trabajo y se espera a que se complete.

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte. Políticas de IAM para servicios integrados

Llamada a Amazon Bedrock con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

Temas

- API de integración de servicios de Amazon Bedrock
- Definición del estado Task para la integración de Amazon Bedrock

API de integración de servicios de Amazon Bedrock

Para integrar AWS Step Functions con Amazon Bedrock, puede utilizar las siguientes API. Estas API son similares a las API de Amazon Bedrock correspondientes, con algunas diferencias en los campos de solicitud que se pasan.

En la siguiente tabla se describen las diferencias entre cada API de integración de servicios y sus API de Amazon Bedrock correspondientes.

API de integración de servicios de Amazon Bedrock y API de Amazon Bedrock correspondientes

API de integración de servicios de Amazon Bedrock	API de Amazon Bedrock correspondiente	Diferencias
InvokeModel Invoca el modelo de Amazon Bedrock especificado para	InvokeModel	El cuerpo de la solicitud de la API de integración de servicios de Amazon

API de integración de servicios de Amazon Bedrock	API de Amazon Bedrock correspondiente	Diferencias
ejecutar la inferencia utilizando la entrada proporcionada en el cuerpo de la solicitud `InvokeModel` sirve para ejecutar la inferencia para modelos de texto, modelos de imagen y modelos de incrustación.		Bedrock incluye los siguientes parámetros adicionales. • `Body`: especifica los datos de entrada en el formato especificado en el encabezado de la solicitud del tipo de contenido. `Body` contiene parámetros específicos del modelo de destino. Si utiliza la API `InvokeModel`, debe especificar el parámetro `Body`. Step Functions no valida la entrada que proporcionada en `Body`. Al especificar `Body` utilizando la integración optimizada de Amazon Bedrock, puede especificar una carga útil de hasta 256 KB. Si su carga útil supera 256 KB, le recomendamos que utilice `Input`. • `Input`: especifica el origen del que se van a recuperar los datos de entrada. Este campo opcional es específico de una integración optimizada de Amazon Bedrock con Step Functions

API de integración de servicios de Amazon Bedrock	API de Amazon Bedrock correspondiente	Diferencias
		. En este campo, puede especificar un `S3Uri`. Puede especificar `Body` en los Parámetros o `Input`, pero no ambos. Si especifica `Input` sin especificar `ContentType` , el tipo de contenido del origen de datos de entrada pasa a ser el valor de `ContentType` . • `Output`: especifica el destino en el que se escribe la respuesta de la API. Este campo opcional es específico de una integración optimizada de Amazon Bedrock con Step Functions . En este campo, puede especificar un `S3Uri`. Si especifica este campo, el cuerpo de la respuesta de la API se sustituye por una referencia a la ubicación de Amazon S3 de la salida original. El siguiente ejemplo muestra la sintaxis de la InvokeModel API para la Amazon Bedrock integración.

API de integración de servicios de Amazon Bedrock	API de Amazon Bedrock correspondiente	Diferencias

		<pre>{
		    "ModelId": String,
		  // required
		    "Accept": String,
		  // default: applicati
		on/json
		    "ContentType":
		 String, // default:
		 application/json
		    "Input": { // not
		 from Bedrock API
		        "S3Uri": String
		    },
		    "Output": { // not
		 from Bedrock API
		        "S3Uri": String
		    }
		}</pre>

CreateModelCustomizationJob Crea un trabajo de microajuste para personalizar un modelo base.	CreateModelCustomizationJob	Ninguna
CreateModelCustomizationJob .sync Crea un trabajo de microajuste para personalizar un modelo base.	CreateModelCustomizationJob	Ninguna

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Definición del estado Task para la integración de Amazon Bedrock

La siguiente definición del estado Task muestra cómo puede integrar Amazon Bedrock en sus máquinas de estado. Este ejemplo muestra un estado Task que extrae el resultado completo de la invocación del modelo especificada en la ruta, `result_one`. Se basa en los parámetros de inferencia de los modelos fundacionales. En este ejemplo se utiliza el modelo de lenguaje grande (LLM) de Cohere Command.

```
{
  "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**
>
> Para implementar un ejemplo de una máquina de estados que se integre con Amazon Bedrock la suya Cuenta de AWS, consulteEncadenamiento de mensajes de IA con Amazon Bedrock.

Llama AWS CodeBuild con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

En qué se diferencia la CodeBuild integración optimizada de la integración del CodeBuild AWS SDK

- Se admite el patrón de integración Ejecutar un trabajo (.sync).
- Después de llamar `StopBuild` o`StopBuildBatch`, la compilación o el lote de compilaciones no se pueden eliminar inmediatamente hasta que se complete algún trabajo interno CodeBuild para finalizar el estado de la compilación o compilaciones. Si se intenta usar `BatchDeleteBuilds` o `DeleteBuildBatch` durante este período, no se podrá eliminar la compilación o el lote de compilación. Las integraciones de servicios optimizados para `BatchDeleteBuilds` y `DeleteBuildBatch` incluyen un reintento interno para simplificar el caso de uso de eliminar justo después de detener.

La integración del AWS Step Functions servicio le AWS CodeBuild permite usar Step Functions para activar, detener y administrar compilaciones, y compartir informes de compilación. Con Step Functions puede diseñar y ejecutar canalizaciones de integración continua para validar los cambios de software para las aplicaciones.

No todas las API admiten todos los patrones de integración, como se muestra en la tabla siguiente.

API	Respuesta de la solicitud	Ejecutar un trabajo (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

> **Los parámetros en Step Functions se expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

CodeBuild API y sintaxis compatibles:

- `StartBuild`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `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

- Sintaxis de la respuesta

- StopBuild
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Id
  - Sintaxis de la respuesta
- BatchDeleteBuilds
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Ids
  - Sintaxis de la respuesta
- BatchGetReports
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - ReportArns
  - Sintaxis de la respuesta
- StartBuildBatch
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - 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

- Sintaxis de la respuesta

- StopBuildBatch
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Id
  - Sintaxis de la respuesta
- RetryBuildBatch
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Id
    - IdempotencyToken
    - RetryType
  - Sintaxis de la respuesta
- DeleteBuildBatch
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Id
  - Sintaxis de la respuesta

> **Note**
>
> Puede usar el operador de descenso recursivo de `JSONPath` (`..`) para `BatchDeleteBuilds`. Esto devuelve una matriz y le permite convertir el campo `Arn` de `StartBuild` en un parámetro `Ids` plural, como se muestra en el siguiente ejemplo.

```
"BatchDeleteBuilds": {
    "Type": "Task",
    "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
    "Parameters": {
        "Ids.$": "$.Build..Arn"
    },
    "Next": "MyNextState"
},
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Llamar a las API de DynamoDB con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> **Note**
>
> Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

> En qué se diferencia la integración optimizada de DynamoDB de la integración del SDK de DynamoDB AWS
>
> - No hay ninguna optimización para el patrón de integración Respuesta de la solicitud.
> - No se admite el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.
> - Solo las acciones de la API `GetItem`, `PutItem`, `UpdateItem` y `DeleteItem` están disponibles a través de la integración optimizada. Otras acciones de la API, como las que `CreateTable`están disponibles mediante la integración del SDK de AWS DynamoDB.

API y sintaxis de Amazon DynamoDB compatibles:

- `GetItem`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `Key`
    - `TableName`
    - `AttributesToGet`

    - ConsistentRead
    - ExpressionAttributeNames
    - ProjectionExpression
    - ReturnConsumedCapacity
  - Sintaxis de la respuesta
- PutItem
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Item
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Sintaxis de la respuesta
- DeleteItem
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity

    - ReturnItemCollectionMetrics
    - ReturnValues
  - Sintaxis de la respuesta
- UpdateItem
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
    - UpdateExpression
  - Sintaxis de la respuesta

> **Los parámetros de se Step Functions expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

A continuación se muestra un estado `Task` que recupera un mensaje de DynamoDB.

```
    "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"
    },
```

Para ver un ejemplo práctico de este estado, consulte el proyecto de muestra Transferencia de registros de datos (Lambda, DynamoDB y Amazon SQS).

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte. Políticas de IAM para servicios integrados

Administrar las tareas de Amazon ECS o Fargate con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de Amazon ECS/Fargate de la integración de Amazon ECS o Fargate SDK AWS
>
> - Se admite el patrón de integración Ejecutar un trabajo (.sync).
> - `ecs:runTask` puede devolver una respuesta HTTP 200, pero tiene un campo `Failures` que no está vacío de la siguiente manera:
>   - Respuesta de solicitud: devuelve la respuesta y no se producen errores en la tarea. Esto es lo mismo que no realizar ninguna optimización.
>   - Ejecutar un trabajo o un token de tarea: si se encuentra un campo `Failures` que no está vacío, se produce un error `AmazonECS.Unknown` en la tarea.

API y sintaxis de ECS/Fargate compatibles:

> **Los parámetros en se expresan en Step Functions PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

- `RunTask` comienza una tarea nueva utilizando la definición de tarea especificada.
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `Cluster`
    - `Group`
    - `LaunchType`
    - `NetworkConfiguration`
    - `Overrides`
    - `PlacementConstraints`
    - `PlacementStrategy`
    - `PlatformVersion`
    - `PropagateTags`
    - `TaskDefinition`
    - `EnableExecuteCommand`
  - Sintaxis de la respuesta

Pasar datos a una tarea de Amazon ECS

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

Puede utilizar `overrides` para anular el comando predeterminado de un contenedor y pasar información de entrada a las tareas de Amazon ECS. Consulte `ContainerOverride`. En el ejemplo, hemos utilizado JsonPath para pasar valores `Task` de la entrada al `Task` estado.

El ejemplo siguiente incluye un estado `Task` que ejecuta una tarea de Amazon ECS y espera a que finalice.

```
{
 "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
    }
  }
}
```

La línea `"Command.$": "$.commands"` en `ContainerOverrides` pasa los comandos de la entrada de estado al contenedor.

En el ejemplo anterior, cada uno de los comandos se transferirá como una anulación del contenedor si la entrada de la ejecución es la siguiente.

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

El ejemplo siguiente incluye un estado `Task` que ejecuta una tarea de Amazon ECS y, a continuación, espera a que se devuelva el token de tarea. Consulte Cómo esperar una devolución de llamada con el token de tarea.

```
{
```

```
   "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
      }
   }
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Llamar a Amazon EKS con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de Amazon EKS de la integración del AWS SDK de Amazon EKS
>
> - Se admite el patrón de integración Ejecutar un trabajo (.sync).

- No hay optimizaciones para el patrón de integración Respuesta de la solicitud.
- No se admite el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Step Functions proporciona dos tipos de API de integración de servicios para integrarse con Amazon Elastic Kubernetes Service. Uno le permite usar las API de Amazon EKS para crear y administrar un clúster de Amazon EKS. El otro le permite interactuar con el clúster mediante la API de Kubernetes y ejecutar trabajos como parte del flujo de trabajo de la aplicación. Puede utilizar las integraciones de la API de Kubernetes con los clústeres de Amazon EKS creados con Step Functions, con los clústeres de Amazon EKS creados con la herramienta eksctl o la consola de Amazon EKS, o métodos similares. Para obtener más información, consulte Creación de un clúster de Amazon EKS en la Guía del usuario de Amazon EKS.

> **Note**
>
> La integración de Step Functions EKS solo admite las API de Kubernetes con acceso a puntos de conexión públicos. De forma predeterminada, los puntos de conexión del servidor API de los clústeres EKS tienen acceso público. Para obtener más información, consulte Control de acceso al punto de conexión del clúster de Amazon EKS en la Guía del usuario de Amazon EKS.

Step Functions no finaliza automáticamente un clúster de Amazon EKS si se detiene la ejecución. Si su máquina de estado se detiene antes de que su clúster de Amazon EKS finalice, es posible que el clúster siga ejecutándose indefinidamente y que se acumulen cargos adicionales. Para evitarlo, asegúrese de que cualquier clúster de Amazon EKS que cree finalice correctamente. Para obtener más información, consulte:

- Eliminación de un clúster en la Guía del usuario de Amazon EKS.
- Ejecutar un trabajo (.sync) en patrones de integración de servicios

> **Note**
>
> Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

Integraciones de API de Kubernetes

Step Functions admite las siguientes API de Kubernetes:

RunJob

La integración del servicio `eks:runJob` le permite ejecutar un trabajo en el clúster de Amazon EKS. Con la variante `eks:runJob.sync`, puede esperar a que se complete el trabajo y, opcionalmente, recuperar los registros.

El servidor de la API de Kubernetes debe conceder permisos al rol de IAM que utiliza la máquina de estado. Para obtener más información, consulte Permisos.

Para el patrón Ejecutar un trabajo (`.sync`), el estado del trabajo se determina mediante un sondeo. Step Functions sondea inicialmente a una velocidad de aproximadamente 1 sondeo por minuto. Con el tiempo, esta velocidad se reduce a aproximadamente 1 encuesta cada 5 minutos. Si necesita sondeos más frecuentes o necesita tener más control sobre la estrategia de sondeo, puede utilizar la integración `eks:call` para consultar el estado del trabajo.

La integración`eks:runJob` es específica de los trabajos de Kubernetes `batch/v1`. Para obtener más información, consulte Trabajos en la documentación de Kubernetes. Si quiere gestionar otros recursos de Kubernetes, incluidos los recursos personalizados, utilice la integración del servicio `eks:call`. Puede usar Step Functions para crear bucles de sondeo, como se ilustra en el proyecto de muestra the section called “Encuesta sobre el estado del trabajo (Lambda,) AWS Batch”.

Entre los parámetros admitidos se incluyen:

- `ClusterName`: el nombre del clúster de Amazon EKS al que desea llamar.
  - `Type`: `String`
  - Obligatorio: sí

- `CertificateAuthority`: los datos de certificados codificados en Base64 necesarios para comunicarse con el clúster. Puede obtener este valor en la consola de Amazon EKS o mediante la DescribeClusterAPI de Amazon EKS.
  - `Type`: `String`
  - Obligatorio: sí
- `Endpoint`: el punto de conexión del servidor de la API de Kubernetes. Puede obtener este valor en la consola de Amazon EKS o mediante la DescribeClusterAPI de Amazon EKS.
  - `Type`: `String`
  - Obligatorio: sí
- `Namespace`: el espacio de nombres en el que se ejecutará el trabajo. Si no se proporciona, se utiliza el espacio de nombres `default`.
  - `Type`: `String`
  - Obligatorio: no
- `Job`: la definición del trabajo de Kubernetes. Consulte Trabajos en la documentación de Kubernetes.
  - `Type`: `JSON` o `String`
  - Obligatorio: sí
- `LogOptions`: un conjunto de opciones para controlar la recuperación opcional de registros. Solo se aplica si se utiliza el patrón de integración del servicio Ejecutar un trabajo (.sync) para esperar a que finalice el trabajo.
  - `Type`: `JSON`
  - Obligatorio: no
  - Los registros se incluyen en la respuesta, debajo de la clave `logs`. Es posible que haya varios pods en la tarea, cada uno con varios contenedores.

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
```

```
        }
      },
      ...
    }
  }
```

  - La recuperación de registros se realiza en la medida de lo posible. Si se produce un error al recuperar un registro, en lugar del campo `log` aparecerán los campos `error` y `cause`.
- `LogOptions.RetrieveLogs`: habilite la recuperación del registro una vez finalizado el trabajo. De forma predeterminada, los registros no se recuperan.
  - `Type`: `Boolean`
  - Obligatorio: no
- `LogOptions.RawLogs`: si `RawLogs` se establece en true, los registros se devolverán como cadenas brutas sin intentar analizarlos para convertirlos en JSON. De forma predeterminada, los registros se deserializan en JSON si es posible. En algunos casos, este análisis puede introducir cambios no deseados, como limitar la precisión de los números que contienen muchos dígitos.
  - `Type`: `Boolean`
  - Obligatorio: no
- `LogOptions.LogParameters`: la API Leer registro de Kubernetes admite parámetros de consulta para controlar la recuperación de registros. Por ejemplo, puede usar `tailLines` o `limitBytes` para limitar el tamaño de los registros recuperados y permanecer dentro de la cuota de tamaño de datos de Step Functions. Para obtener más información, consulte sección Leer registro de la referencia de la API de Kubernetes.
  - `Type:` Mapa de `String` a `List of Strings`
  - Obligatorio: no
  - Ejemplo:

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

En el siguiente ejemplo se incluye un estado `Task` que ejecuta un trabajo, espera a que se complete y, posteriormente, recupera los registros del trabajo:

```
{
  "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`

La integración del servicio `eks:call` le permite usar la API de Kubernetes para leer y escribir objetos de recursos de Kubernetes a través de un punto de conexión de la API de Kubernetes.

El servidor de la API de Kubernetes debe conceder permisos al rol de IAM que utiliza la máquina de estado. Para obtener más información, consulte Permisos.

Para obtener más información sobre estas operaciones disponibles, consulte la Referencia de la API de Kubernetes.

Entre los parámetros admitidos para `Call` se incluyen:

- `ClusterName`: el nombre del clúster de Amazon EKS al que desea llamar.
  - `Type`: cadena
  - Obligatorio: sí
- `CertificateAuthority`: los datos de certificados codificados en Base64 necesarios para comunicarse con el clúster. Puede obtener este valor en la consola de Amazon EKS o mediante la DescribeClusterAPI de Amazon EKS.
  - `Type`: `String`
  - Obligatorio: sí
- `Endpoint`: el punto de conexión del servidor de la API de Kubernetes. Puede encontrar este valor en la consola de Amazon EKS o mediante la DescribeCluster API de Amazon EKS.
  - `Type`: `String`
  - Obligatorio: sí
- `Method`: el método HTTP de su solicitud. Puede ser uno de los siguientes: `GET`, `POST`, `PUT`, `DELETE`, `HEAD` o `PATCH`.
  - `Type`: `String`
  - Obligatorio: sí
- `Path`: la ruta HTTP de la operación de la API de REST de Kubernetes.
  - `Type`: `String`
  - Obligatorio: sí

- `QueryParameters`: los parámetros de consulta HTTP de la operación de la API de REST de Kubernetes.
  - `Type: Mapa de String a List of Strings`
  - Obligatorio: no
  - Ejemplo:

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- `RequestBody`: el cuerpo del mensaje HTTP de la operación de la API de REST de Kubernetes.
  - `Type: JSON o String`
  - Obligatorio: no

El ejemplo siguiente incluye un estado `Task` que utiliza `eks:call` para enumerar los pods que pertenecen al trabajo `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
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que utiliza `eks:call` para eliminar el trabajo `example-job` y establece la `propagationPolicy` para garantizar que también se eliminan los pods del trabajo.

```
{
  "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
    }
  }
}
```

API de Amazon EKS compatibles

Las API y la sintaxis de Amazon EKS compatibles incluyen:

- `CreateCluster`
  - Sintaxis de la solicitud
  - Sintaxis de la respuesta

    Cuando se crea un clúster de Amazon EKS mediante la integración del servicio `eks:createCluster`, el rol de IAM se añade a la tabla de autorización de RBAC de Kubernetes como administrador (con permisos system:masters). Inicialmente, solo esa entidad de IAM puede realizar llamadas al servidor de la API de Kubernetes. Para obtener más información, consulte:

- Administración de usuarios o roles de IAM para su clúster en la Guía del usuario de Amazon EKS.
- La sección Permisos

Amazon EKS utiliza roles vinculados a un servicio que contienen los permisos que Amazon EKS necesita para llamar a otros servicios en su nombre. Si estos roles vinculados a servicios aún no existen en su cuenta, debe añadir el permiso `iam:CreateServiceLinkedRole` al rol de IAM que utiliza Step Functions. Para obtener más información, consulte Uso de roles vinculados a servicios en la Guía del usuario de Amazon EKS.

El rol de IAM utilizado por Step Functions debe tener permisos `iam:PassRole` para transferir el rol de IAM del clúster a Amazon EKS. Para obtener más información, consulte Rol de IAM de clúster de Amazon EKS en la Guía del usuario de Amazon EKS.

- `DeleteCluster`
  - Sintaxis de la solicitud
  - Sintaxis de la respuesta

  Debe eliminar todos los perfiles o grupos de nodos de Fargate antes de eliminar un clúster.
- `CreateFargateProfile`
  - Sintaxis de la solicitud
  - Sintaxis de la respuesta

  Amazon EKS utiliza roles vinculados a un servicio que contienen los permisos que Amazon EKS necesita para llamar a otros servicios en su nombre. Si estos roles vinculados a servicios aún no existen en su cuenta, debe añadir el permiso `iam:CreateServiceLinkedRole` al rol de IAM que utiliza Step Functions. Para obtener más información, consulte Uso de roles vinculados a servicios en la Guía del usuario de Amazon EKS.

  Es posible que Amazon EKS o Fargate no estén disponibles en todas las regiones. Para obtener información sobre la disponibilidad de la región, consulte la sección sobre Fargate en la Guía del usuario de Amazon EKS.

  El rol de IAM utilizado por Step Functions debe tener permisos `iam:PassRole` para transferir el rol de IAM de la ejecución de pods a Amazon EKS. Para obtener más información, consulte Rol de ejecución de pod en la guía del usuario de Amazon EKS.
- `DeleteFargateProfile`

  - Sintaxis de la solicitud
  - Sintaxis de la respuesta
- `CreateNodegroup`
  - Sintaxis de la solicitud
  - Sintaxis de la respuesta

    Amazon EKS utiliza un rol vinculado al servicio que contiene los permisos que este necesita para llamar a otros servicios de en su nombre. Si estos roles vinculados a servicios aún no existen en su cuenta, debe añadir el permiso `iam:CreateServiceLinkedRole` al rol de IAM que utiliza Step Functions. Para obtener más información, consulte Uso de roles vinculados a servicios en la Guía del usuario de Amazon EKS.

    El rol de IAM utilizado por Step Functions debe tener permisos `iam:PassRole` para transferir el rol de IAM del nodo a Amazon EKS. Para obtener más información, consulte Uso de roles vinculados a servicios en la Guía del usuario de Amazon EKS.
- `DeleteNodegroup`
  - Sintaxis de la solicitud
  - Sintaxis de la respuesta

El ejemplo siguiente incluye una `Task` que crea un clúster de Amazon EKS.

```
{
  "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
```

```
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que elimina un clúster de Amazon EKS.

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
      },
      "End": true
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que crea un perfil de Fargate.

```
{
  "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
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que elimina un perfil de Fargate.

```
{
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que crea un grupo de nodos.

```
{
  "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
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que elimina un grupo de nodos.

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup"
      },
      "End": true
    }
  }
}
```

Permisos

Cuando se crea un clúster de Amazon EKS mediante la integración del servicio `eks:createCluster`, el rol de IAM se añade a la tabla de autorización de RBAC de Kubernetes como administrador, con permisos `system:masters`. Inicialmente, solo esa entidad de IAM puede realizar llamadas al servidor de la API de Kubernetes. Por ejemplo, no se podrá usar kubectl para interactuar con el servidor de API de Kubernetes, a menos que se asuma el mismo rol que su máquina de estado de Step Functions o si configura Kubernetes para conceder permisos a entidades de IAM adicionales. Para obtener más información, consulte Administración de usuarios o roles de IAM para su clúster en la Guía del usuario de Amazon EKS.

Puede añadir permisos para entidades de IAM adicionales, como usuarios o roles, añadiéndolas al `aws-auth ConfigMap` en el espacio de nombres kube-system. Si va a crear el clúster a partir de Step Functions, utilice la integración del servicio `eks:call`.

El ejemplo siguiente incluye un estado `Task` que crea un `aws-auth ConfigMap` y otorga un permiso `system:masters` al usuario `arn:aws:iam::123456789012:user/my-user` y al rol de IAM `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**
>
> Puede que se vea el ARN de un rol de IAM en un formato que incluya la ruta /service-role/, como `arn:aws:iam::123456789012:role/service-role/my-role`. Este token de ruta service-role no debe incluirse al enumerar el rol en `aws-auth`.

Cuando el clúster se crea por primera vez, `aws-auth ConfigMap` no existirá, sino que se agregará automáticamente si crea un perfil de Fargate. Puede recuperar el valor actual de `aws-auth`, añadir los permisos adicionales y PUT una nueva versión. Por lo general, es más sencillo crear `aws-auth` antes que el perfil de Fargate.

Si el clúster se creó fuera de Step Functions, puede configurar kubectl para que se comunique con su servidor de API de Kubernetes. A continuación, se crea un nuevo `aws-auth ConfigMap` mediante `kubectl apply -f aws-auth.yaml` o se edita uno que ya exista mediante `kubectl edit -n kube-system configmap/aws-auth`. Para obtener más información, consulte:

- Crear un kubeconfig para Amazon EKS en la Guía del usuario de Amazon EKS.
- Administración de usuarios o roles de IAM para su clúster en la Guía del usuario de Amazon EKS.

Si el rol de IAM no tiene permisos suficientes en Kubernetes, las integraciones de servicios `eks:call` o `eks:runJob` generarán el siguiente error:

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
    "kind": "Status",
    "apiVersion": "v1",
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
    "code": 401
  },
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

Llamar a Amazon EMR con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de Amazon EMR de la integración del SDK de Amazon EMR AWS
>
> La integración del servicio Amazon EMR optimizado tiene un conjunto personalizado de API que agrupa las API de Amazon EMR subyacentes, tal como se describe a continuación. Por ello, difiere considerablemente de la integración del servicio Amazon EMR AWS SDK. Además, se admite el patrón de integración Ejecutar un trabajo (.sync).

Para la integración AWS Step Functions con Amazon EMR, utiliza las API de integración de servicios de Amazon EMR proporcionadas. Las API de integración de servicios son similares a las API de

Amazon EMR correspondientes, con algunas diferencias en los campos que se pasan y en las respuestas que se devuelven.

Step Functions no finaliza automáticamente un clúster de Amazon EMR si se detiene la ejecución. Si su máquina de estado se detiene antes de que su clúster de Amazon EMR finalice, es posible que el clúster siga ejecutándose indefinidamente y que se acumulen cargos adicionales. Para evitarlo, asegúrese de que cualquier clúster de Amazon EMR que cree finalice correctamente. Para obtener más información, consulte:

- Control de la terminación de los clústeres en la guía del usuario de Amazon EMR.
- La sección Ejecutar un trabajo (.sync) de Patrones de integración de servicios.

> **Note**
>
> A partir de `emr-5.28.0`, puede especificar el parámetro `StepConcurrencyLevel` al crear un clúster para permitir que diferentes pasos se ejecuten en paralelo en un único clúster. Puede utilizar los estados `Map` y `Parallel` de Step Functions para enviar trabajo en paralelo al clúster.

La disponibilidad de la integración de servicios de Amazon EMR depende de la disponibilidad de las API de Amazon EMR. Consulte la documentación de Amazon EMR para conocer las limitaciones en regiones especiales.

> **Note**
>
> Para su integración con Amazon EMR, Step Functions tiene una frecuencia de sondeo de trabajo codificada de 60 segundos durante los primeros 10 minutos y de 300 segundos después.

En la siguiente tabla se describen las diferencias entre cada API de integración de servicios y sus API de Amazon EMR correspondientes.

API de integración de servicios de Amazon EMR y las correspondientes API de Amazon EMR

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias
createCluster Crea y comienza a ejecutar un clúster (flujo de trabajo). Amazon EMR está vinculado directamente a un tipo de rol de IAM único conocido como rol vinculado a servicio. Para que `createCluster` y `createCluster.sync` funcionen, tiene que tener configurados los permisos necesarios para crear el rol vinculado a servicios `AWSServiceRoleForEMRCleanup`. Para obtener más información al respecto, incluida una instrucción que puede añadir a la política de permisos de IAM, consulte Uso del rol vinculado a servicios para Amazon EMR.	runJobFlow	`createCluster` utiliza la misma sintaxis de solicitud que runJobFlow, excepto en lo siguiente: • El campo `Instances.KeepJobFlowAliveWhenNoSteps` es obligatorio y debe tener el valor booleano `TRUE`. • El campo `Steps` no está permitido. • El campo `Instances.InstanceFleets[index].Name` se debe proporcionar y debe ser único si la API de conector `modifyInstanceFleetByName` opcional se utiliza. • El campo `Instances.InstanceGroups[index].Name` se debe proporcionar y debe ser único si la API `modifyInstanceGroupByName` opcional se utiliza. La respuesta es la siguiente: `{ "ClusterId": "string"`

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias

		<pre>}</pre> Amazon EMR utiliza lo siguiente: <pre>{
		  "JobFlowId": "string"
		}</pre>

createCluster.sync Crea y comienza a ejecutar un clúster (flujo de trabajo).	runJobFlow	Lo mismo que `createCluster` , pero espera a que el clúster alcance el estado `WAITING`.

setClusterTerminationProtección
 	
Bloquea un clúster (flujo de trabajo) de forma que las instancias de EC2 del clúster no se puedan terminar mediante la intervención del usuario, una llamada a la API o un error del flujo de trabajo.
	setTerminationProtection	La solicitud utiliza: <pre>{
		  "ClusterId": "string"
		}</pre> Amazon EMR utiliza lo siguiente: <pre>{
		  "JobFlowIds":
		 ["string"]
		}</pre>

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias

terminateCluster
 	
Cierra un clúster (flujo de trabajo).
	terminateJobFlows	La solicitud utiliza:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR utiliza lo siguiente:
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster.sync Cierra un clúster (flujo de trabajo).	terminateJobFlows	Lo mismo que `terminateCluster` , pero espera a que el clúster finalice.

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias

addStep
 	
Agrega un nuevo paso a un clúster en ejecución.
 	
Si lo desea, también puede especificar el parámetro `ExecutionRoleArn` mientras se utiliza esta API.
	addJobFlowPasos	La solicitud utiliza la clave `"ClusterId"` . Amazon EMR utiliza `"JobFlowId"` . La solicitud utiliza un solo paso.
		
		`{ "Step": <"StepConfig object"> }`
		
		Amazon EMR utiliza lo siguiente:
		
		`{ "Steps": [<StepConfig objects>] }`
		
		La respuesta es la siguiente:
		
		`{ "StepId": "string" }`
		
		Amazon EMR devuelve lo siguiente:
		
		`{ "StepIds": [<strings>] }`

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias
addStep.sync Agrega un nuevo paso a un clúster en ejecución. Si lo desea, también puede especificar el parámetro `ExecutionRoleArn` mientras se utiliza esta API.	addJobFlowPasos	Lo mismo que `addStep`, pero espera a que el paso se complete.

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias

cancelStep
 	
Cancela un paso pendiente en un clúster en ejecución.
	cancelSteps	La solicitud utiliza:
		
		`{ "StepId": "string" }`
		
		Amazon EMR utiliza lo siguiente:
		
		`{ "StepIds": [<strings>] }`
		
		La respuesta es la siguiente:
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		Amazon EMR utiliza lo siguiente:
		
		`{ "CancelStepsInfoList": [<CancelStepsInfo objects>] }`

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias
modifyInstanceFleetByName Modifica las capacidades de Spot de destino y de destino bajo demanda para la flota de instancias con el `InstanceFleetName` especificado.	modifyInstanceFleet	La solicitud es la misma que para `modifyInstanceFleet`, excepto por lo siguiente: • El campo `Instance.InstanceFleetId` no está permitido. • En el tiempo de ejecución, el `InstanceFleetId` lo determina automáticamente la integración del servicio mediante una llamada a `ListInstanceFleets` y un análisis del resultado.

API de integración de servicios de Amazon EMR	API de EMR correspondiente	Diferencias

modifyInstanceGroupByName
 	
Modifica el número de nodos y las opciones de configuración de un grupo de instancias.
	modifyInstanceGroups	La solicitud es la siguiente:
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		Amazon EMR utiliza una lista:
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		En el objeto `InstanceGroupModifyConfig`, el campo `InstanceGroupId` no está permitido.
		
		Se ha añadido un nuevo campo, `InstanceGroupName`. En el tiempo de ejecución, el `InstanceGroupId` lo determina automáticamente la integración del servicio mediante una llamada a `ListInstanceGroups` y un análisis del resultado.

El código siguiente incluye un estado `Task` que crea un clúster.

```
"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
```

```
}
```

El código siguiente incluye un estado `Task` que habilita la protección de la terminación.

```
"Enable_Termination_Protection": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

El código siguiente incluye un estado `Task` que envía un paso a un clúster.

```
"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
```

```
}
```

El código siguiente incluye un estado `Task` que cancela un paso.

```
"Cancel_Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

El código siguiente incluye un estado `Task` que termina un clúster.

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

El código siguiente incluye un estado `Task` que escala un clúster hacia arriba o hacia abajo para un grupo de instancias.

```
"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

El código siguiente incluye un estado `Task` que escala un clúster hacia arriba o hacia abajo para una flota de instancias.

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Llame a Amazon EMR en EKS con AWS Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de Amazon EMR en EKS de la integración de Amazon EMR en el SDK de EKS AWS
>
> - Se admite el patrón de integración Ejecutar un trabajo (.sync).
> - No hay optimizaciones para el patrón de integración Respuesta de la solicitud.
> - No se admite el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.

> **Note**
>
> Para su integración con Amazon EMR, Step Functions tiene una frecuencia de sondeo de trabajo codificada de 60 segundos durante los primeros 10 minutos y de 300 segundos después.

Para realizar la integración AWS Step Functions con Amazon EMR en EKS, utilice las API de integración de servicios de Amazon EMR en EKS. Las API de integración de servicios son las mismas que las correspondientes API de Amazon EMR en EKS, pero no todas las API admiten todos los patrones de integración, como se muestra en la siguiente tabla.

API	Respuesta de la solicitud	Ejecutar un trabajo (.sync)
CreateVirtualCluster	✓	
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

API de Amazon EMR en EKS compatibles:

> **Note**
>
> Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

- `CreateVirtualCluster`
  - Sintaxis de la solicitud
  - Parámetros admitidos
  - Sintaxis de la respuesta
- `DeleteVirtualCluster`
  - Sintaxis de la solicitud

- Parámetros admitidos
- Sintaxis de la respuesta
- `StartJobRun`
  - Sintaxis de la solicitud
  - Parámetros admitidos
  - Sintaxis de la respuesta

El ejemplo siguiente incluye un estado `Task` que crea un clúster virtual.

```
"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
}
```

El ejemplo siguiente incluye un estado `Task` que envía un trabajo a un clúster virtual y espera a que finalice.

```
"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
}
```

El ejemplo siguiente incluye un estado `Task` que elimina un clúster virtual y espera a que finalice.

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
```

```
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte. Políticas de IAM para servicios integrados

Llamada a Amazon EMR Serverless con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> Diferencia entre la integración optimizada de EMR Serverless y la integración del SDK de EMR Serverless AWS
>
> - La integración del servicio de EMR Serverless optimizado tiene un conjunto personalizado de API que agrupan las API de EMR Serverless subyacentes. Debido a esta personalización, la EMR Serverless integración optimizada difiere considerablemente de la integración de los servicios del EMR Serverless AWS SDK. Además, la integración de EMR Serverless optimizada admite el patrón de integración Ejecutar un trabajo (.sync).
> - No se admite el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.

En este tema

API de integración de servicios de EMR Serverless

Para integrar AWS Step Functions con EMR Serverless, puede utilizar las siguientes seis API de integración de servicios de EMR Serverless. Estas API de integración de servicios son similares a las API de EMR Serverless correspondientes, con algunas diferencias en los campos que se pasan y en las respuestas que se devuelven.

En la siguiente tabla se describen las diferencias entre cada API de integración de servicios y sus API de EMR Serverless correspondientes.

API de integración de servicios de EMR Serverless y API de EMR Serverless correspondientes

API de integración de servicios de EMR Serverless	API de EMR Serverless correspondiente	Diferencias
createApplication Crea una aplicación. EMR Serverless está vinculado a un tipo de rol de IAM único conocido como rol vinculado a servicio. Para que `createApplication` y `createApplication.sync` funcionen, tiene que tener configurados los permisos necesarios para crear el rol vinculado a servicios `AWS ServiceRoleForAmazonEMRServerless` . Para obtener más información al respecto, incluida una instrucción que puede añadir a la política de permisos de IAM, consulte Uso de roles vinculados a servicios para EMR Serverless.	CreateApplication	Ninguna
createApplication.sync Crea una aplicación.	CreateApplication	No hay diferencias entre las solicitudes y las respuestas de la API de EMR Serverless y la API de integración de servicios de EMR Serverless. Sin embargo, createApp

API de integración de servicios de EMR Serverless	API de EMR Serverless correspondiente	Diferencias
		lication.sync espera a que la aplicación alcance el estado `CREATED`.
startApplication Inicia una aplicación especific ada e inicializa la capacidad inicial de la aplicación si está configurada.	StartApplication	La respuesta de la API de EMR Serverless no contiene ningún dato, pero la respuesta de la API de integración de servicio EMR Serverless incluye los siguientes datos. `{ "ApplicationId": "string" }`
startApplication.sync Inicia una aplicación específic a e inicializa la capacidad inicial si está configurada.	StartApplication	La respuesta de la API de EMR Serverless no contiene ningún dato, pero la respuesta de la API de integración de servicio EMR Serverless incluye los siguientes datos. `{ "ApplicationId": "string" }` Además, startApplication.s ync espera a que la aplicación alcance el estado `STARTED`.

API de integración de servicios de EMR Serverless	API de EMR Serverless correspondiente	Diferencias
stopApplication Detiene una aplicación especificada y libera la capacidad inicial si está configurada. Todos los trabajos programados y en ejecución deben completarse o cancelarse antes de detener una aplicación.	StopApplication	La respuesta de la API de EMR Serverless no contiene ningún dato, pero la respuesta de la API de integración de servicio EMR Serverless incluye los siguientes datos. `{ "ApplicationId": "string" }`
stopApplication.sync Detiene una aplicación especificada y libera la capacidad inicial si está configurada. Todos los trabajos programados y en ejecución deben completarse o cancelarse antes de detener una aplicación.	StopApplication	La respuesta de la API de EMR Serverless no contiene ningún dato, pero la respuesta de la API de integración de servicio EMR Serverless incluye los siguientes datos. `{ "ApplicationId": "string" }` Además, stopApplication.sync espera a que la aplicación alcance el estado `STOPPED`.

API de integración de servicios de EMR Serverless	API de EMR Serverless correspondiente	Diferencias
deleteApplication Elimina una aplicación. Para poder eliminarla, una aplicación debe estar en el estado `STOPPED` o `CREATED`.	DeleteApplication	La respuesta de la API de EMR Serverless no contiene ningún dato, pero la respuesta de la API de integración de servicio EMR Serverless incluye los siguientes datos. `{ "ApplicationId": "string" }`
deleteApplication.sync Elimina una aplicación. Para poder eliminarla, una aplicación debe estar en el estado `STOPPED` o `CREATED`.	DeleteApplication	La respuesta de la API de EMR Serverless no contiene ningún dato, pero la respuesta de la API de integración de servicio EMR Serverless incluye los siguientes datos. `{ "ApplicationId": "string" }` Además, stopApplication.sync espera a que la aplicación alcance el estado `TERMINATED` .
startJobRun Inicia una ejecución de trabajo.	StartJobRun	Ninguna

API de integración de servicios de EMR Serverless	API de EMR Serverless correspondiente	Diferencias
startJobRun.sync Inicia una ejecución de trabajo.	StartJobRun	No hay diferencias entre las solicitudes y las respuestas de la API de EMR Serverless y la API de integración de servicios de EMR Serverless. Sin embargo, startJobRun.sync espera a que la aplicación alcance el estado. `SUCCESS`
cancelJobRun Cancela una ejecución de trabajo.	CancelJobRun	Ninguna
cancelJobRun.sync Cancela una ejecución de trabajo.	CancelJobRun	No hay diferencias entre las solicitudes y las respuestas de la API de EMR Serverless y la API de integración de servicios de EMR Serverless. Sin embargo, cancelJobRun.sync espera a que la aplicación alcance el estado. `CANCELLED`

Casos de uso de integración de EMR sin servidor

Para la integración optimizada del servicio de EMR Serverless, se recomienda crear una sola aplicación y, a continuación, utilizarla para ejecutar varios trabajos. Por ejemplo, en una máquina de un solo estado, puede incluir varias startJobRunsolicitudes, todas las cuales utilizan la misma aplicación. En los siguientes ejemplos de estados Estado de la tarea se muestran casos de uso con los que integrar API de EMR Serverless con Step Functions Para obtener información sobre otros casos de uso de EMR Serverless, consulte Qué es Amazon EMR Serverless.

> **Tip**
>
> Para implementar un ejemplo de una máquina de estados que se integre y ejecute varios trabajos en su computadora Cuenta de AWS, consulteEjecutar un trabajo de EMR Serverless. EMR Serverless

- Crear una aplicación
- Iniciar una aplicación
- Detener una aplicación
- Eliminar una aplicación
- Iniciar un trabajo en una aplicación
- Cancelar un trabajo en una aplicación

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

En los ejemplos que se muestran en los siguientes casos de uso, sustituya el texto *en cursiva* por la información específica del recurso. Por ejemplo, *yourApplicationId*sustitúyalo por el ID de tu EMR Serverless aplicación, por ejemplo`00yv7iv71inak893`.

Crear una aplicación

En el siguiente ejemplo de estado Task se crea una aplicación mediante la API de integración del servicio CreateApplication.sync.

```
"Create_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:createApplication.sync",
    "Parameters": {
        "Name": "MyApplication",
        "ReleaseLabel": "emr-6.9.0",
        "Type": "SPARK"
    },
    "End": true
}
```

Iniciar una aplicación

En el siguiente ejemplo de estado Task se inicia una aplicación mediante la API de integración del servicio StartApplication.sync.

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Detener una aplicación

En el siguiente ejemplo de estado Task se detiene una aplicación mediante la API de integración del servicio StopApplication.sync.

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Eliminar una aplicación

En el siguiente ejemplo de estado Task se elimina una aplicación mediante la API de integración del servicio DeleteApplication.sync.

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Iniciar un trabajo en una aplicación

El siguiente ejemplo de estado de tarea inicia un trabajo en una aplicación mediante la API de integración de servicios startJobRun.sync.

```
"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
}
```

Cancelar un trabajo en una aplicación

En el siguiente ejemplo de estado de tarea, se cancela un trabajo en una aplicación mediante la API de integración de servicios cancelJobRun.sync.

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

Llama EventBridge con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

En qué se diferencia la EventBridge integración optimizada de la integración del EventBridge AWS SDK

- El ARN de ejecución y el ARN de la máquina de estado se añaden automáticamente al campo `Resources` de cada `PutEventsRequestEntry`.
- Si la respuesta de `PutEvents` contiene un `FailedEntryCount` distinto de cero, el estado `Task` genera el error `EventBridge.FailedEntry`.

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Step Functions proporciona una API de integración de servicios para la integración con Amazon EventBridge. Esto le permite crear aplicaciones basadas en eventos al enviar eventos personalizados directamente desde los flujos de trabajo de Step Functions.

Para usar la `PutEvents` API, tendrás que crear una EventBridge regla en tu cuenta que coincida con el patrón específico de los eventos que vas a enviar. Por ejemplo, puede:

- Cree una función Lambda en su cuenta que reciba e imprima un evento que coincida con una EventBridge regla.
- Cree una EventBridge regla en su cuenta en el bus de eventos predeterminado que coincida con un patrón de eventos específico y se dirija a la función Lambda.

Para obtener más información, consulte:

- Añadir EventBridge eventos de Amazon PutEvents en la Guía EventBridge del usuario.
- Cómo esperar una devolución de llamada con el token de tarea en patrones de integración de servicios

> **Note**
>
> Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

EventBridge API compatible

La EventBridge API y la sintaxis compatibles incluyen:

- `PutEvents`
  - Sintaxis de la solicitud
  - Parámetro admitido:
    - `Entries`
  - Sintaxis de la respuesta

El siguiente ejemplo incluye una `Task` que envía un evento personalizado:

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

Control de errores

La API de `PutEvents` acepta una matriz de entradas como entrada y, a continuación, devuelve una matriz de entradas de resultados. Siempre y cuando la acción `PutEvents` se haya realizado correctamente, `PutEvents` devolverá una respuesta HTTP 200, incluso si una o varias entradas generaron un error. `PutEvents` devuelve el número de entradas con error en el campo `FailedEntryCount`.

Step Functions comprueba si `FailedEntryCount` es mayor que cero. Si es mayor que cero, Step Functions genera el error `EventBridge.FailedEntry`. Esto le permite usar el control de errores integrado de Step Functions en los estados de las tareas para detectar o reintentar entradas con error, en lugar de tener que usar un estado adicional para analizar el `FailedEntryCount` de la respuesta.

> **Note**
>
> Si ha implementado la idempotencia y puede reintentarlo de forma segura en todas las entradas, puede utilizar la lógica de reintento de Step Functions. Step Functions no elimina las entradas correctas de la matriz de entradas `PutEvents` antes de volver a intentarlo. En su lugar, lo vuelve a intentar con la matriz original de entradas.

Gestione AWS Glue trabajos con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> **En qué se diferencia la AWS Glue integración optimizada de la integración del AWS GlueAWS SDK**
>
> - El patrón de integración Ejecutar un trabajo (.sync) está disponible.
> - El campo `JobName` se extrae de la solicitud y se inserta en la respuesta, que normalmente solo contiene `JobRunID`.

AWS Glue API compatible:

- `StartJobRun`

> **Los parámetros en Step Functions se expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

A continuación se incluye un `Task` estado que inicia un AWS Glue trabajo.

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
      "Next": "ValidateOutput"
    },
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte Políticas de IAM para servicios integrados.

Gestione AWS Glue DataBrew trabajos con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

Puede utilizar la DataBrew integración para añadir pasos de limpieza y normalización de datos a sus flujos de trabajo de análisis y aprendizaje automático.

DataBrew API compatible:

- `StartJobRun`

A continuación se incluye un `Task` estado que inicia un trabajo de solicitud-respuesta DataBrew.

```
"DataBrew StartJobRun": {
```

```
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
            "Parameters": {
               "Name": "sample-proj-job-1"
            },
            "Next": "NEXT_STATE"
          },
```

A continuación se incluye un `Task` estado que inicia un trabajo de sincronización DataBrew .

```
"DataBrew StartJobRun": {
           "Type": "Task",
           "Resource": "arn:aws:states:::databrew:startJobRun.sync",
           "Parameters": {
              "Name": "sample-proj-job-1"
           },
           "Next": "NEXT_STATE"
          },
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Invocar Lambda con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde [Lenguaje de estados de Amazon](#) (ASL). Para obtener más información, consulte [Trabajo con otros servicios](#) y [Cómo pasar parámetros a una API de servicio](#).

> En qué se diferencia la integración optimizada de Lambda de la integración del SDK de Lambda AWS
>
> - El campo `Payload` de la respuesta se analiza de Json de escape a Json.
> - Si la respuesta contiene el campo `FunctionError` o se genera una excepción en la función de Lambda, se produce un error en la tarea.

Para obtener más información acerca de cómo administrar la entrada, la salida y los resultados de los estados, consulte [Procesamiento de entrada y salida en Step Functions](#).

APIs compatibles AWS Lambda :

- `Invoke`
  - Sintaxis de la solicitud
  - Parámetros admitidos
    - `ClientContext`
    - `FunctionName`
    - `InvocationType`
    - `Qualifier`
    - `Payload`
  - Sintaxis de la respuesta

> **Los parámetros de Step Functions se expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

El ejemplo siguiente incluye un estado `Task` que invoca una función de Lambda.

```
{
   "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
      }
   }
}
```

El ejemplo siguiente incluye un estado `Task` que implementa el patrón de integración de servicios de devolución de llamada.

```
{
   "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
      }
   }
}
```

Al invocar una función de Lambda, la ejecución esperará a que se complete la función. Si invoca la función de Lambda con una tarea de devolución de llamada, el tiempo de espera del latido no comienza a contar hasta que la función de Lambda haya terminado de ejecutarse y haya devuelto un resultado. Siempre y cuando se ejecute la función de Lambda, no se aplicará el tiempo de espera de latido.

También es posible llamar a Lambda de forma asíncrona mediante el parámetro `InvocationType`, como se muestra en el siguiente ejemplo:

> **Note**
>
> Para las invocaciones asíncronas de funciones de Lambda, el período de tiempo de espera del latido comienza de inmediato.

```
{

  "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
    }
  }
}
```

Cuando se devuelve el resultado `Task`, el resultado de la función se anida dentro de un diccionario de metadatos. Por ejemplo:

```
{

   "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
}
```

Como alternativa, puede invocar una función de Lambda especificando el ARN de una función directamente en el campo «Recurso». Cuando se invoca una función de Lambda de esta manera, no se puede especificar `.waitForTaskToken` y el resultado de la tarea solo contiene el resultado de la función.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

Puede invocar un alias o una versión de la función de Lambda específica mediante la introducción de estas opciones en el ARN, en el campo `Resource`. Consulte los siguientes temas en la documentación de Lambda:

- Control de versiones de AWS Lambda
- AWS Lambda alias

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Administre AWS Elemental MediaConvert con Step Functions

> **Experimenta con Step Functions y MediaConvert**
>
> Aprenda a utilizar la integración MediaConvert optimizada en un flujo de trabajo que detecta y elimina las barras de color SMTPE de longitud desconocida del principio de un videoclip. Lea la entrada del blog del 12 de abril de 2024: Flujos de trabajo con poco código con AWS Elemental MediaConvert

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> **En qué se diferencia la integración optimizada de la integración estándar AWS del SDK**
>
> - El patrón de integración Ejecutar un trabajo (.sync) está disponible.

- Sin optimizaciones Respuesta de la solicitud ni patrones de Cómo esperar una devolución de llamada con el token de tarea integración.

MediaConvert APIs compatibles:

- `CreateJob`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `Role` (Obligatorio)
    - `Settings` (Obligatorio)
    - `CreateJobRequest` (Opcional)
  - Sintaxis de respuesta: consulte CreateJobResponse el esquema

A continuación se incluye un `Task` estado en el que se envía un MediaConvert trabajo y se espera a que se complete.

```
{
    "StartAt": "MediaConvert_CreateJob",
    "States": {
        "MediaConvert_CreateJob": {
        "Type": "Task",
        "Resource": "arn:aws:states:::mediaconvert:createJob.sync",
        "Parameters": {
            "Role": "arn:aws:iam::111122223333:role/Admin",
            "Settings": {
            "OutputGroups": [
                {
                "Outputs": [
                    {
                    "ContainerSettings": {
                        "Container": "MP4"
                    },
                    "VideoDescription": {
                        "CodecSettings": {
                        "Codec": "H_264",
                        "H264Settings": {
                            "MaxBitrate": 1000,
                            "RateControlMode": "QVBR",
```

```
                            "SceneChangeDetect": "TRANSITION_DETECTION"
                        }
                        }
                    },
                    "AudioDescriptions": [
                        {
                        "CodecSettings": {
                            "Codec": "AAC",
                            "AacSettings": {
                            "Bitrate": 96000,
                            "CodingMode": "CODING_MODE_2_0",
                            "SampleRate": 48000
                            }
                        }
                        }
                    ]
                    }
                ],
                "OutputGroupSettings": {
                    "Type": "FILE_GROUP_SETTINGS",
                    "FileGroupSettings": {
                    "Destination": "s3://DOC-EXAMPLE-DESTINATION-BUCKET/"
                    }
                }
                }
            ],
            "Inputs": [
                {
                "AudioSelectors": {
                    "Audio Selector 1": {
                    "DefaultSelection": "DEFAULT"
                    }
                },
                "FileInput": "s3://DOC-EXAMPLE-SOURCE-BUCKET/DOC-EXAMPLE-SOURCE_FILE"
                }
            ]
            }
        },
        "End": true
        }
    }
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utiliza Step Functions con MediaConvert, consulte. Políticas de IAM para AWS Elemental MediaConvert

> **Los parámetros en Step Functions se expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

Administre SageMaker con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> **En qué se diferencia la SageMaker integración optimizada de la integración del SageMaker AWS SDK**
>
> - Se admite el patrón de integración Ejecutar un trabajo (.sync).
> - No hay optimizaciones para el patrón de integración Respuesta de la solicitud.
> - No se admite el patrón de integración Cómo esperar una devolución de llamada con el token de tarea.

SageMaker API y sintaxis compatibles:

- `CreateEndpoint`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `EndpointConfigName`
    - `EndpointName`
    - `Tags`
  - Sintaxis de la respuesta
- `CreateEndpointConfig`
  - Sintaxis de la solicitud

- Parámetros admitidos:
  - EndpointConfigName
  - KmsKeyId
  - ProductionVariants
  - Tags
- Sintaxis de la respuesta

- CreateHyperParameterTuningJob

> **Note**
>
> Esta acción de la API es compatible con el patrón de integración de .sync.

  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - HyperParameterTuningJobConfig
    - HyperParameterTuningJobName
    - Tags
    - TrainingJobDefinition
    - WarmStartConfig
  - Sintaxis de la respuesta
- CreateLabelingJob

> **Note**
>
> Esta acción de la API es compatible con el patrón de integración de .sync.

  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - HumanTaskConfig
    - InputConfig
    - LabelAttributeName

    - LabelCategoryConfigS3Uri
    - LabelingJobAlgorithmsConfig
    - LabelingJobName
    - OutputConfig
    - RoleArn
    - StoppingConditions
    - Tags
  - Sintaxis de la respuesta
- CreateModel
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - Containers
    - EnableNetworkIsolation
    - ExecutionRoleArn
    - ModelName
    - PrimaryContainer
    - Tags
    - VpcConfig
- CreateProcessingJob

> **Note**
>
> Esta acción de la API es compatible con el patrón de integración de `.sync`.

  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - AppSpecification
    - Environment
    - ExperimentConfig
    - NetworkConfig
    - ProcessingInputs

    - ProcessingJobName
    - ProcessingOutputConfig
    - ProcessingResources
    - RoleArn
    - StoppingCondition
    - Tags
  - Sintaxis de la respuesta
- CreateTrainingJob

> **Note**
>
> Esta acción de la API es compatible con el patrón de integración de .sync.

  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - AlgorithmSpecification
    - HyperParameters
    - InputDataConfig
    - OutputDataConfig
    - ResourceConfig
    - RoleArn
    - StoppingCondition
    - Tags
    - TrainingJobName
    - VpcConfig
  - Sintaxis de la respuesta
- CreateTransformJob

> **Note**
>
> Esta acción de la API es compatible con el patrón de integración de .sync.

> **Note**
>
> AWS Step Functions no creará automáticamente una política para`CreateTransformJob`. Debe asociar una política insertada al rol que se ha creado. Para obtener más información, consulte esta política de IAM de ejemplo: `CreateTrainingJob`.

- Sintaxis de la solicitud
- Parámetros admitidos:
  - `BatchStrategy`
  - `Environment`
  - `MaxConcurrentTransforms`
  - `MaxPayloadInMB`
  - `ModelName`
  - `Tags`
  - `TransformInput`
  - `TransformJobName`
  - `TransformOutput`
  - `TransformResources`
- Sintaxis de la respuesta

- `UpdateEndpoint`
  - Sintaxis de la solicitud
  - Parámetros admitidos:
    - `EndpointConfigName`
    - `EndpointName`
  - Sintaxis de la respuesta

SageMaker Ejemplo de Transform Job

A continuación se incluye un `Task` estado que crea un trabajo de SageMaker transformación de Amazon y especifica la ubicación de Amazon S3 para `DataSource` y`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 Ejemplo de trabajo de formación

A continuación se incluye un `Task` estado que crea un puesto de SageMaker formación en Amazon.

```
{
   "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 Ejemplo de trabajo de etiquetado

A continuación se incluye un `Task` estado que crea un trabajo de SageMaker etiquetado de Amazon.

```
{
  "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 Ejemplo de trabajo de procesamiento

A continuación se incluye un `Task` estado que crea un trabajo de SageMaker procesamiento de Amazon.

```
{
  "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."
    }
  }
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Llamar a Amazon SNS con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de Amazon SNS de la integración del SDK de Amazon AWS SNS
>
> No hay optimizaciones para los patrones de integración Respuesta de la solicitud o Cómo esperar una devolución de llamada con el token de tarea.

API admitidas de Amazon SNS:

> Note
>
> Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

- `Publish`
  - Sintaxis de la solicitud
  - Parámetros admitidos
    - `Message`
    - `MessageAttributes`
    - `MessageStructure`
    - `PhoneNumber`
    - `Subject`
    - `TargetArn`
    - `TopicArn`

- Sintaxis de la respuesta

> **Los parámetros de se Step Functions expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

El ejemplo siguiente incluye un estado `Task` que publica en un tema de Amazon Simple Notification Service (Amazon SNS).

```
{
 "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
    }
  }
}
```

Pasar valores dinámicos. Puede modificar el ejemplo anterior para pasar dinámicamente un atributo de esta carga de JSON:

```
{
"input": {
  "message": "Hello world"
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

Anexa `.$` al campo `StringValue`:

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

El siguiente ejemplo incluye un estado `Task` que publica en un tema de Amazon SNS y, a continuación, espera a que se devuelva el token de tarea. Consulte Cómo esperar una devolución de llamada con el token de tarea.

```
{
   "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
      }
   }
```

```
}
```

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte. Políticas de IAM para servicios integrados

Llamar a Amazon SQS con Step Functions

Step Functions puede controlar ciertos AWS servicios directamente desde Lenguaje de estados de Amazon (ASL). Para obtener más información, consulte Trabajo con otros servicios y Cómo pasar parámetros a una API de servicio.

> En qué se diferencia la integración optimizada de Amazon SQS de la integración del SDK de Amazon AWS SQS
>
> No hay optimizaciones para los patrones de integración Respuesta de la solicitud o Cómo esperar una devolución de llamada con el token de tarea.

API admitidas de Amazon SQS:

> Note
>
> Hay una cuota para el tamaño máximo de los datos de entrada o resultado para una tarea en Step Functions. Esto limita a 256 KB de datos como cadena codificada en UTF-8 al enviar o recibir datos de otro servicio. Consulte Cuotas relacionadas con ejecuciones de máquinas de estado.

- `SendMessage`

  Parámetros admitidos:

  - `DelaySeconds`
  - `MessageAttribute`
  - `MessageBody`
  - `MessageDeduplicationId`
  - `MessageGroupId`
  - `QueueUrl`

- Response syntax

> **Los parámetros de se Step Functions expresan en PascalCase**
>
> Incluso si la API del servicio nativo está en CamelCase, por ejemplo, la `startSyncExecution` acción de la API, se especifican parámetros PascalCase en, como:. `StateMachineArn`

El ejemplo siguiente incluye un estado `Task` que envía un mensaje de Amazon Simple Queue Service (Amazon SQS).

```
{
 "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
    }
  }
}
```

El ejemplo siguiente incluye un estado `Task` que publica en una cola de Amazon SQS y, a continuación, espera a que se devuelva el token de tarea. Consulte Cómo esperar una devolución de llamada con el token de tarea.

```
{
   "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
      }
   }
}
```

Para obtener más información acerca de la recepción de mensajes en Amazon SQS, consulte Recepción y eliminación del mensaje en la Guía para desarrolladores de Amazon Simple Queue Service.

Para obtener información sobre cómo configurar IAM los permisos cuando se utilizan Step Functions con otros AWS servicios, consulte. Políticas de IAM para servicios integrados

Gestione AWS Step Functions las ejecuciones como un servicio integrado

Step Functions se integra con su propia API como integración de servicios. Esto permite a Step Functions iniciar una nueva ejecución de una máquina de estado directamente desde el estado de la tarea de una ejecución en curso. Al crear nuevos flujos de trabajo, utilice las ejecuciones de flujos de trabajo anidados para reducir la complejidad de los flujos de trabajo principales y para reutilizar los procesos comunes.

En qué se diferencia la integración optimizada de Step Functions de la integración del AWS SDK de Step Functions

- El patrón de integración Ejecutar un trabajo (.sync) está disponible.

Tenga en cuenta que no hay optimizaciones para los patrones de integración Respuesta de la solicitud o Cómo esperar una devolución de llamada con el token de tarea.

Para obtener más información, consulte los siguientes temas:

- Inicio de ejecuciones desde una tarea
- Trabajo con otros servicios
- Cómo pasar parámetros a una API de servicio

API y sintaxis admitidas de Step Functions:

- `StartExecution`
  - Sintaxis de la solicitud
  - Parámetros admitidos
    - `Input`
    - `Name`
    - `StateMachineArn`
  - Sintaxis de la respuesta

El ejemplo siguiente incluye un estado `Task` que inicia una ejecución de otra máquina de estado y espera a que se complete.

```
{
   "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
```

```
}
```

El ejemplo siguiente incluye un estado `Task` que inicia una ejecución de otra máquina de estado.

```
{
   "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
}
```

El ejemplo siguiente incluye un estado `Task` que implementa el patrón de integración de servicios de devolución de llamada.

```
{
   "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
}
```

Para asociar una ejecución de flujo de trabajo anidado a la ejecución principal que la inició, transfiera un parámetro denominado especialmente que incluya el ID de ejecución extraído del objeto context. Al iniciar una ejecución anidada, utilice un parámetro denominado `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID`. Transfiera el ID de ejecución añadiendo `.`

$ al nombre del parámetro y haciendo referencia al ID en el objeto context con `$$.Execution.Id`. Para obtener más información, consulte Acceso al objeto de contexto.

```
{
   "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
}
```

Las máquinas de estado anidadas devuelven lo siguiente:

Recurso	Salida
startExecution.sync	Cadena
startExecution.sync:2	JSON

Ambos esperarán a que se complete la máquina de estado anidada, pero devuelven diferentes formatos `Output`. Por ejemplo, si crea una función de Lambda que devuelve el objeto `{ "MyKey": "MyValue" }`, obtendría las siguientes respuestas:

Para startExecution.sync:

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

Para startExecution.sync:2:

```
{
```

```
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

Configuración de los permisos de IAM para máquinas de estado anidadas

Una máquina de estado principal determina si una máquina de estado secundaria ha completado la ejecución mediante sondeos y eventos. Las votaciones requieren permiso `states:DescribeExecution` mientras que los eventos enviados EventBridge a Step Functions requieren permisos para`events:PutTargets`,`events:PutRule`, y`events:DescribeRule`. Si faltan estos permisos en su rol de IAM, es posible que se produzca un retraso antes de que una máquina de estado principal se dé cuenta de que se ha completado la ejecución de la máquina de estado secundaria.

Para una máquina de estado que llame a `StartExecution` para una única ejecución de flujo de trabajo anidado, utilice una política de IAM que limite los permisos a esa máquina de estado.

Para obtener más información, consulte Permisos de IAM para Step Functions.

Llamada a API de terceros

Una tarea HTTP es un tipo de estado de Estado de la tarea que permite llamar a cualquier API pública de terceros, como Salesforce y Stripe, en los flujos de trabajo. Para llamar a una API de terceros, utilice el estado Task con el recurso de `arn:aws:states:::http:invoke`. A continuación, proporcione los detalles de configuración del punto de conexión de la API, como la URL de la API, el método que desea utilizar y los detalles de autenticación.

Si utiliza Workflow Studio para crear una máquina de estado que contenga una tarea HTTP, Workflow Studio generará automáticamente un rol de ejecución con políticas de IAM para la tarea HTTP. Para obtener más información, consulte Rol para probar tareas HTTP en Workflow Studio.

Temas

- Definición de tarea HTTP
- Campos de tareas HTTP
- Autenticación de una tarea HTTP
- Combinación de datos de conexión de EventBridge y definición de tarea HTTP

- Aplicación de codificación URL en el cuerpo de la solicitud
- Permisos de IAM para ejecutar una tarea HTTP
- Ejemplo de tarea HTTP
- Probar una tarea HTTP
- Respuestas a tareas HTTP no admitidas

Definición de tarea HTTP

La definición de ASL representa una tarea HTTP con un recurso `http:invoke`. La siguiente definición de tarea HTTP invoca una API de Stripe que devuelve una lista de todos los clientes.

```
"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
}
```

Campos de tareas HTTP

Una tarea HTTP incluye los siguientes campos en su definición.

`Resource` (Obligatorio)

Para especificar un tipo de tarea, indique su ARN en el campo `Resource`. Para una tarea HTTP, especifique el campo `Resource` de la siguiente manera.

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (Obligatorio)

Contiene los campos `ApiEndpoint`, `Method` y `ConnectionArn` que proporcionan información sobre la API de terceros a la que quiere llamar. `Parameters` también contiene campos opcionales, como `Headers` y `QueryParameters`.

Puede especificar una combinación de JSON estático y JsonPathsintaxis como `Parameters` en el `Parameters` campo. Para obtener más información, consulte Cómo pasar parámetros a una API de servicio.

`ApiEndpoint`(Obligatorio)

Especifica la URL de la API de terceros a la que desea llamar. Para añadir parámetros de consulta a la URL, use el campo `QueryParameters`. En el ejemplo siguiente se muestra cómo se puede llamar a una API de Stripe para recuperar la lista de todos los clientes.

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

También puedes especificar una ruta de referencia mediante la JsonPathsintaxis para seleccionar el nodo JSON que contiene la URL de la API de terceros. Por ejemplo, supongamos que desea llamar a una de las API de Stripe con un ID de cliente específico. Supongamos que ha proporcionado la siguiente entrada de estado.

```
{
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

Para recuperar los detalles de este ID de cliente mediante una API de Stripe, especifique el `ApiEndpoint` como se muestra en el ejemplo siguiente. En este ejemplo se utiliza una función intrínseca y una ruta de referencia.

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

En tiempo de ejecución, Step Functions resuelve el valor de `ApiEndpoint` de la siguiente manera.

```
https://api.stripe.com/v1/customers/1234567890
```

`Method`(Obligatorio)

Especifica el método HTTP que desea utilizar para llamar a una API de terceros. Puede especificar uno de estos métodos en su tarea HTTP: GET, POST, PUT, DELETE, PATCH, OPTIONS o HEAD.

Por ejemplo, para usar el método GET, especifique el campo `Method` de la siguiente manera.

```
"Method": "GET"
```

También puede usar una ruta de referencia para especificar el método en tiempo de ejecución. Por ejemplo, **`"Method.$": "$.myHTTPMethod"`**.

`Authentication`(Obligatorio)

Contiene el campo `ConnectionArn` que especifica cómo autenticar una llamada a la API de terceros. Step Functions admite autenticación de un `ApiEndpoint` especificado mediante el recurso de conexión de Amazon EventBridge.

`ConnectionArn`(Obligatorio)

Especifica el ARN de conexión de EventBridge.

Una tarea HTTP requiere una EventBridge conexión que gestione de forma segura las credenciales de autenticación de un proveedor de API. Una conexión especifica el tipo de autorización y las credenciales que se van a utilizar para autorizar una API de terceros. El uso de una conexión ayuda a evitar secretos de codificación rígida, como claves de API, en la definición de la máquina de estado. En una conexión, también puede especificar los parámetros `Headers`, `QueryParameters` y `RequestBody`.

Al crear una EventBridge conexión, proporciona sus detalles de autenticación. Para obtener más información acerca de cómo funciona la autenticación para una tarea HTTP, consulte Autenticación de una tarea HTTP.

En el ejemplo siguiente se muestra cómo se puede especificar el campo `Authentication` en la definición de tarea HTTP.

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
}
```

`Headers` (Opcional)

Proporciona contexto y metadatos adicionales al punto de conexión de la API. Puede especificar encabezados como cadena o como matriz JSON.

Puede especificar encabezados en la conexión de EventBridge y el campo `Headers` en una tarea HTTP. Le recomendamos que no incluya en el campo `Headers` detalles de autenticación de sus proveedores de API. Le recomendamos que incluya estos detalles en su conexión de EventBridge.

Step Functions agrega los encabezados que especifique en la conexión de EventBridge a los encabezados que especifique en la definición de tarea HTTP. Si la definición y la conexión contienen las mismas claves de encabezado, Step Functions utiliza los valores correspondientes especificados en la conexión de EventBridge para esos encabezados. Para obtener más información sobre cómo Step Functions realiza combinación de datos, consulte Combinación de datos de conexión de EventBridge y definición de tarea HTTP.

El siguiente ejemplo especifica un encabezado que se incluirá en una llamada a la API de terceros: `content-type`.

```
"Headers": {
  "content-type": "application/json"
}
```

También puede utilizar una ruta de referencia para especificar los encabezados en tiempo de ejecución. Por ejemplo, **`"Headers.$": "$.myHTTPHeaders"`**.

Step Functions establece los encabezados `User-Agent`, `Range` y `Host`. Step Functions establece el valor del encabezado `Host` en función de la API a la que esté llamando. A continuación se muestra un ejemplo de estos encabezados.

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

No puede usar los siguientes encabezados en su definición de tarea HTTP. Si utiliza estos encabezados, la tarea HTTP producirá el error `States.Runtime`.

- A-IM
- Accept-Charset

- Accept-Datetime
- Accept-Encoding
- Cache-Control
- Connection
- Content-Encoding
- Content-MD5
- Date
- Expect
- Forwarded
- De
- Host
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Origen
- Pragma
- Proxy-Authorization
- Referer
- Server
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Advertencia

- x-forwarded-*
- x-amz-*
- x-amzn-*

`QueryParameters` (Opcional)

Inserta pares de clave-valor al final de la URL de una API. Puede especificar los parámetros de consulta como cadena, matriz JSON u objeto JSON. Step Functions codifica automáticamente en URL los parámetros de consulta cuando llama a una API de terceros.

Por ejemplo, supongamos que desea llamar a la API de Stripe para buscar clientes que realicen sus transacciones en dólares estadounidenses (USD). Supongamos que ha proporcionado los siguientes `QueryParameters` como entrada de estado.

```
"QueryParameters": {
  "currency": "usd"
}
```

En tiempo de ejecución, Step Functions añade los `QueryParameters` al URL de la API de la siguiente manera.

```
https://api.stripe.com/v1/customers/search?currency=usd
```

También puede usar una ruta de referencia para especificar los parámetros de consulta en tiempo de ejecución. Por ejemplo, **`"QueryParameters.$": "$.myQueryParameters"`**.

Si especificó los parámetros de consulta en su conexión de EventBridge, Step Functions agrega estos parámetros de consulta a los parámetros de consulta que especifique en la definición de tarea HTTP. Si la definición y la conexión contienen los mismos parámetros de consulta, Step Functions utiliza los valores correspondientes especificados en la conexión de EventBridge para esos encabezados. Para obtener más información sobre cómo Step Functions realiza combinación de datos, consulte Combinación de datos de conexión de EventBridge y definición de tarea HTTP.

`Transform` (Opcional)

Contiene los campos `RequestBodyEncoding` y `RequestEncodingOptions`. De forma predeterminada, Step Functions envía el cuerpo de la solicitud como datos JSON a un punto de conexión de la API.

Si su proveedor de API acepta los cuerpos de la solicitud `form-urlencoded`, utilice el campo `Transform` para especificar codificación URL de los cuerpos de la solicitud. También debe especificar el encabezado `content-type` como `application/x-www-form-urlencoded`. A continuación, Step Functions codifica en URL automáticamente el cuerpo de la solicitud.

`RequestBodyEncoding`

Especifica la codificación URL del cuerpo de la solicitud. Puede especificar los valores siguientes: `NONE` o `URL_ENCODED`.

- `NONE`: el cuerpo de la solicitud HTTP será el JSON serializado del campo `RequestBody`. Este es el valor predeterminado.
- `URL_ENCODED`: el cuerpo de la solicitud HTTP serán los datos del formulario codificado en URL del campo `RequestBody`.

`RequestEncodingOptions`

Determina la opción de codificación que se utilizará para las matrices en el cuerpo de la solicitud si establece `RequestBodyEncoding` como `URL_ENCODED`.

Step Functions admite las siguientes opciones de codificación de matrices. Para obtener más información sobre estas opciones y ejemplos, consulte Aplicación de codificación URL en el cuerpo de la solicitud.

- `INDICES`: codifica matrices utilizando el valor de índice de los elementos de la matriz. De forma predeterminada, Step Functions utiliza esta opción de codificación.
- `REPEAT`: repite una clave para cada elemento de una matriz.
- `COMMAS`: codifica todos los valores de una clave como una lista de valores delimitada por comas.
- `BRACKETS`: repite una clave para cada elemento de una matriz y añade un corchete, [], a la clave para indicar que se trata de una matriz.

El siguiente ejemplo establece codificación URL para los datos del cuerpo de la solicitud. También especifica el uso de la opción de codificación `COMMAS` para matrices en el cuerpo de la solicitud.

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
```

```
  }
}
```

`RequestBody` (Opcional)

Acepta los datos JSON que se proporcionan en la entrada de estado. En`RequestBody`, puedes especificar una combinación de JSON estático y JsonPathsintaxis. Por ejemplo, supongamos que proporciona la siguiente entrada de estado:

```
{
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

Para utilizar estos valores de `CardNumber` y `ExpiryDate` en el cuerpo de la solicitud en tiempo de ejecución, puede especificar los siguientes datos de JSON en el cuerpo de la solicitud.

```
"RequestBody": {
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

Si la API de terceros a la que desea llamar requiere cuerpos de la solicitud `form-urlencoded`, debe especificar codificación URL para los datos del cuerpo de la solicitud. Para obtener más información, consulte Aplicación de codificación URL en el cuerpo de la solicitud.

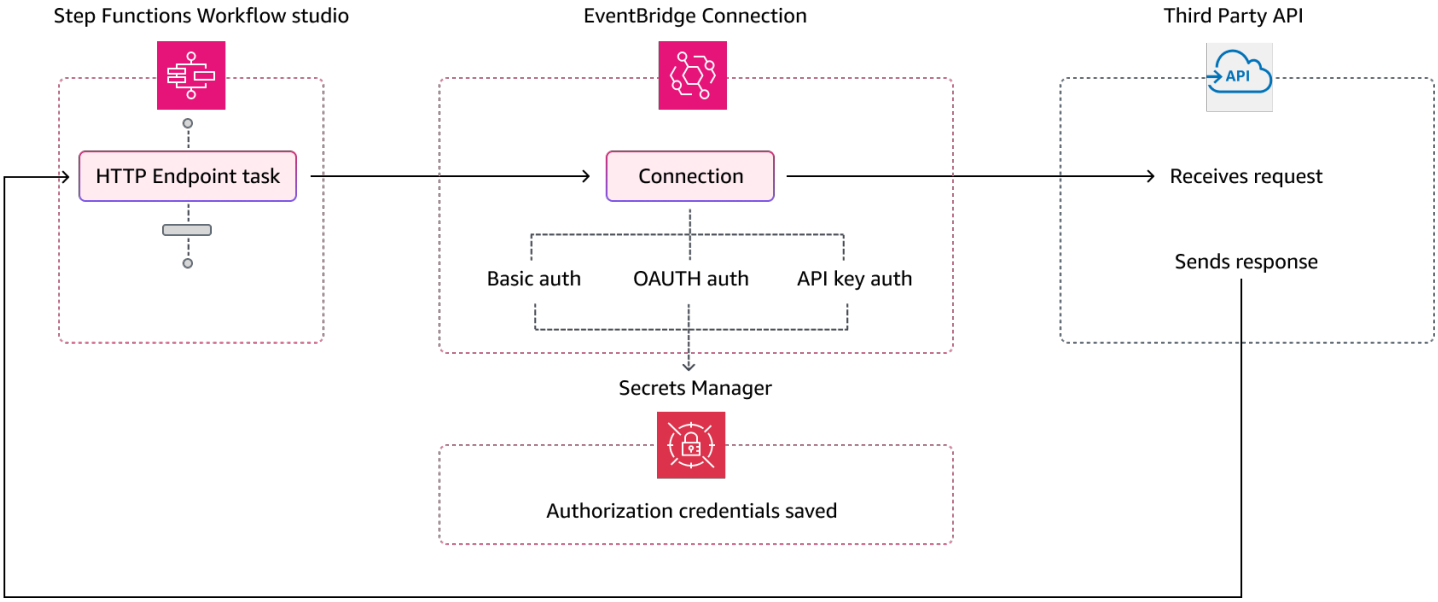
Autenticación de una tarea HTTP

Una tarea HTTP requiere una EventBridge conexión que gestione de forma segura las credenciales de autenticación de un proveedor de API. Una conexión especifica el tipo de autorización y las credenciales que se van a utilizar para autorizar una API de terceros. El uso de una conexión ayuda a evitar secretos de codificación rígida, como claves de API, en la definición de la máquina de estado. Una EventBridge conexión admite los esquemas de autorización Basic, OAuth y API Key.

Al crear una EventBridge conexión, proporciona sus detalles de autenticación. También puede incluir el encabezado, el cuerpo y los parámetros de consulta necesarios para la autorización con una API. Debe incluir el ARN de conexión en cualquier tarea HTTP que llame a una API de terceros.

Al crear una conexión y añadir parámetros de autorización, EventBridge crea una entrada secreta AWS Secrets Manager. En este secreto, EventBridge almacena los parámetros de conexión y autorización de forma cifrada. Para crear o actualizar correctamente una conexión, debe usar una persona Cuenta de AWS que tenga permiso para usar Secrets Manager. Para obtener más información sobre los IAM permisos que su máquina estatal necesita para acceder a una EventBridge conexión, consultePermisos de IAM para ejecutar una tarea HTTP.

La siguiente imagen muestra cómo gestiona Step Functions la autenticación de las llamadas a la API de terceros mediante una conexión de EventBridge.



Combinación de datos de conexión de EventBridge y definición de tarea HTTP

Al invocar una tarea HTTP puede especificar datos en la conexión de EventBridge y la definición de tarea HTTP. Estos datos incluyen parámetros `Headers`, `QueryParameters` y `RequestBody`. Antes de llamar a una API de terceros, Step Functions combina el cuerpo de la solicitud con los parámetros del cuerpo de la conexión en todos los casos, salvo si el cuerpo de la solicitud es una cadena y los parámetros del cuerpo de conexión no están vacíos. En este caso, la tarea HTTP produce un error `States.Runtime`.

Si hay claves duplicadas especificadas en la definición de la tarea HTTP y la conexión de EventBridge, Step Functions sobrescribe los valores de la tarea HTTP con los valores de la conexión.

En la siguiente lista se describe cómo combina Step Functions los datos antes de llamar a una API de terceros:

- Encabezados: Step Functions agrega los encabezados que especificó en la conexión a los encabezados del campo `Headers` de la tarea HTTP. Si hay un conflicto entre las claves de los encabezados, Step Functions utiliza los valores especificados en la conexión para esos encabezados. Por ejemplo, si especificó el encabezado `content-type` tanto en la definición de tarea HTTP como en la conexión de EventBridge, Step Functions utiliza el valor del encabezado `content-type` especificado en la conexión.
- Parámetros de consulta: Step Functions agrega cualquier parámetro de consulta que haya especificado en la conexión a los parámetros de consulta del campo `QueryParameters` de la tarea HTTP. Si hay un conflicto entre las claves de los parámetros de consulta, Step Functions utiliza los valores especificados en la conexión para esos encabezados de consulta. Por ejemplo, si especificó el parámetro de consulta `maxItems` tanto en la definición de tarea HTTP como en la conexión de EventBridge, Step Functions utiliza el valor del parámetro de consulta `maxItems` especificado en la conexión.
- Body parameters (Parámetros del cuerpo)
  - Step Functions agrega los valores del cuerpo de la solicitud especificados en la conexión al cuerpo de la solicitud en el campo `RequestBody` de la tarea HTTP. Si hay un conflicto entre las claves de los encabezados de la solicitud, Step Functions utiliza los valores especificados en la conexión para esos encabezados. Por ejemplo, supongamos que especificó un campo `Mode` en el `RequestBody` de la definición de tarea HTTP y en la conexión de EventBridge. Step Functions utiliza el valor del campo `Mode` especificado en la conexión.
  - Si especifica el cuerpo de la solicitud como cadena en lugar de como objeto JSON y la conexión de EventBridge también contiene el cuerpo de la solicitud, Step Functions no puede combinar el cuerpo de la solicitud especificado en ambos lugares. La tarea HTTP produce el error `States.Runtime`.

  Step Functions aplica todas las transformaciones y serializa el cuerpo de la solicitud después de completar la combinación del cuerpo de la solicitud.

El siguiente ejemplo establece los campos `Headers`, `QueryParameters` y `RequestBody` en la tarea HTTP y en la conexión de EventBridge.

Definición de tarea HTTP

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

Conexión de EventBridge

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

En este ejemplo se especifican claves duplicadas en la tarea HTTP y en la conexión de EventBridge. Por tanto, Step Functions sobrescribe los valores de la tarea HTTP con los valores de la conexión. El siguiente fragmento de código muestra la solicitud HTTP que envía Step Functions a la API de terceros.

```
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"}
```

Aplicación de codificación URL en el cuerpo de la solicitud

De forma predeterminada, Step Functions envía el cuerpo de la solicitud como datos JSON a un punto de conexión de la API. Si su proveedor de API externo requiere cuerpos de la solicitud `form-urlencoded`, debe especificar codificación URL para los cuerpos de la solicitud. A continuación, Step Functions codifica en URL automáticamente el cuerpo de la solicitud basándose en la opción de codificación de URL que seleccione.

La codificación URL se especifica mediante el campo `Transform`. Este campo contiene el campo `RequestBodyEncoding` que especifica si desea aplicar o no la codificación URL a los cuerpos de la solicitud. Al especificar el campo `RequestBodyEncoding`, Step Functions convierte el cuerpo de la solicitud JSON en el cuerpo de la solicitud `form-urlencoded` antes de llamar a la API de terceros. También debe especificar el encabezado `content-type` como `application/x-www-form-urlencoded` porque las API que aceptan datos codificados en URL esperan el encabezado `content-type`.

Para codificar matrices en el cuerpo de la solicitud, Step Functions proporciona las siguientes opciones de codificación de matrices.

- `INDICES`: repite una clave para cada elemento de una matriz y añade un corchete, [], a la clave para indicar que se trata de una matriz. Este paréntesis contiene el índice del elemento de la matriz. Añadir el índice ayuda a especificar el orden de los elementos de la matriz. De forma predeterminada, Step Functions utiliza esta opción de codificación.

  Por ejemplo, si el cuerpo de la solicitud contiene la siguiente matriz.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions codifica esta matriz en la siguiente cadena.

  ```
  array[0]=a&array[1]=b&array[2]=c&array[3]=d
  ```

- `REPEAT`: repite una clave para cada elemento de una matriz.

  Por ejemplo, si el cuerpo de la solicitud contiene la siguiente matriz.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions codifica esta matriz en la siguiente cadena.

```
array=a&array=b&array=c&array=d
```

- `COMMAS`: codifica todos los valores de una clave como una lista de valores delimitada por comas.

  Por ejemplo, si el cuerpo de la solicitud contiene la siguiente matriz.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions codifica esta matriz en la siguiente cadena.

  ```
  array=a,b,c,d
  ```

- `BRACKETS`: repite una clave para cada elemento de una matriz y añade un corchete, [], a la clave para indicar que se trata de una matriz.

  Por ejemplo, si el cuerpo de la solicitud contiene la siguiente matriz.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functions codifica esta matriz en la siguiente cadena.

  ```
  array[]=a&array[]=b&array[]=c&array[]=d
  ```

Permisos de IAM para ejecutar una tarea HTTP

Su rol de ejecución de la máquina estatal debe tener los permisos `states:InvokeHTTPEndpoint`, `events:RetrieveConnectionCredentials`, `secretsmanager:GetSecretValue` y `secretsmanager:DescribeSecret` para que una tarea HTTP pueda llamar a una API de terceros. El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios para el rol de la máquina de estado para llamar a las API de Stripe. Esta política de IAM también concede permiso al rol de máquina de estado para acceder a una EventBridge conexión específica, incluido el secreto de esta conexión que está almacenado en Secrets Manager.

```
{
    "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/*"
        }
    ]
}
```

Ejemplo de tarea HTTP

La siguiente definición de máquina de estado muestra una tarea HTTP que incluye los parámetros `Headers`, `QueryParameters`, `Transform` y `RequestBody`. La tarea HTTP llama a una API de Stripe, https://api.stripe.com/v1/invoices, para generar una factura. La tarea HTTP también especifica la codificación URL del cuerpo de la solicitud mediante la opción de codificación `INDICES`.

Asegúrese de que ha creado una conexión de EventBridge. En el ejemplo siguiente se muestra una conexión creada con el tipo de autenticación BASIC.

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

Recuerde reemplazar el texto en *cursiva* por la información específica del recurso.

```
{
  "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
    }
  }
}
```

Para ejecutar esta máquina de estado, proporcione el ID del cliente como entrada, tal como se muestra en el ejemplo siguiente:

```
{
    "customer_id": "1234567890"
}
```

El ejemplo siguiente muestra la solicitud HTTP que Step Functions envía a la API de Stripe.

```
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
```

Probar una tarea HTTP

Puede usar la TestStateAPI a través de la consola, el SDK o el AWS CLI para probar una tarea HTTP. El siguiente procedimiento describe cómo usar la TestState API en la Step Functions consola. Puede probar de forma iterativa la solicitud, la respuesta y los detalles de autenticación de la API hasta que la tarea HTTP funcione según lo esperado.

Prueba del estado Task de HTTP en la consola de Step Functions

1. Abra la consola de Step Functions.
2. Seleccione Crear máquina de estado para empezar a crear una máquina de estado o elija una máquina de estado existente que contenga una tarea HTTP.

   Consulte el paso 4 si está probando la tarea en una máquina de estado existente.
3. En el Modo Diseño de Workflow Studio, configure una tarea HTTP de forma visual. O elija el modo Código para copiar y pegar la definición de la máquina de estado desde su entorno de desarrollo local.
4. En modo Diseño, elija Probar estado en el panel de Inspector de Workflow Studio.
5. En el cuadro de diálogo Probar estado, haga lo siguiente:

   a. En Rol de ejecución, elija un rol de ejecución para probar el estado. Si no tiene un rol con permisos suficientes para una tarea HTTP, consulte Rol para probar tareas HTTP en Workflow Studio para crear un rol.

   b. (Opcional) Proporcione cualquier entrada JSON que necesite el estado seleccionado para la prueba.

   c. En Nivel de inspección, mantenga la selección predeterminada de INFO. Este nivel muestra el estado de la llamada a la API y la salida del estado. Resulta útil para comprobar rápidamente la respuesta de la API.

   d. Seleccione Iniciar prueba.

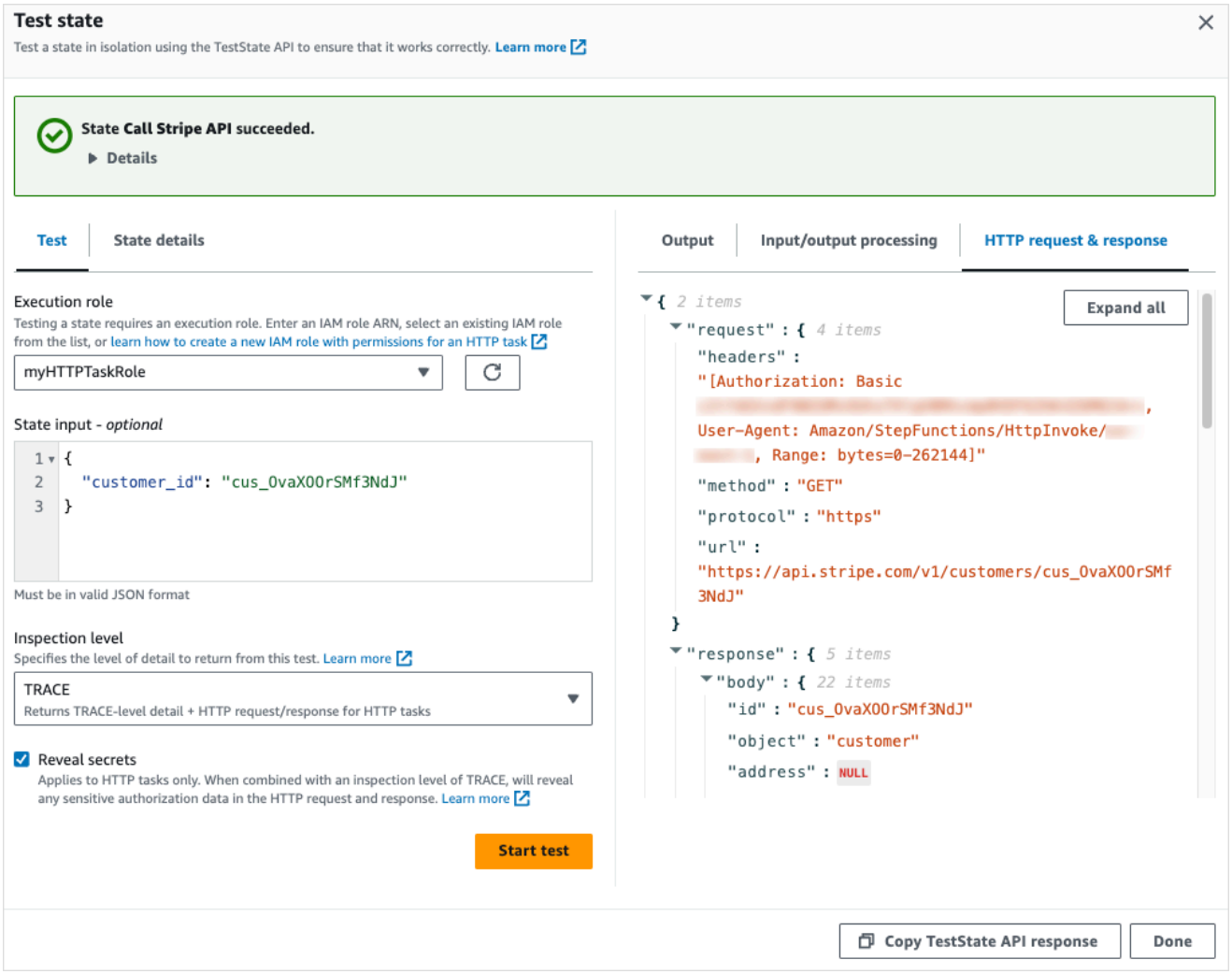
e. Si la prueba se realiza correctamente, el resultado del estado aparece en el lado derecho del cuadro de diálogo Probar estado. Si la prueba no se realiza correctamente, aparece un error.

   En la pestaña Detalles del estado del cuadro de diálogo puede ver la definición del estado y un enlace a la conexión de EventBridge.

f. Cambie el Nivel de inspección a TRACE. Este nivel muestra la solicitud y la respuesta HTTP sin procesar y es útil para verificar encabezados, parámetros de consulta y otros detalles específicos de la API.

g. Seleccione la casilla Revelar secretos. En combinación con TRACE, esta configuración permite ver los datos confidenciales que inserta la conexión de EventBridge, como claves de API. La identidad del usuario de IAM que utilice para acceder a la consola debe tener permiso para realizar la acción `states:RevealSecrets`. Sin este permiso, Step Functions produce un error de acceso denegado al iniciar la prueba. Para ver una política de IAM de ejemplo que establece el permiso `states:RevealSecrets`, consulte IAMpermisos para usar la TestState API.

   En la siguiente imagen se muestra una prueba para una tarea HTTP que se ha realizado correctamente. El Nivel de inspección de este estado se establece en TRACE. La pestaña Solicitud y respuesta HTTP de la siguiente imagen muestra el resultado de la llamada a la API de terceros.



h. Seleccione Iniciar prueba.

i. Si la prueba se realiza correctamente, puede ver sus detalles HTTP en la pestaña Solicitud y respuesta HTTP.

Respuestas a tareas HTTP no admitidas

Una tarea HTTP produce un error `States.Runtime` si se cumple alguna de las siguientes condiciones para la respuesta devuelta:

- La respuesta contiene un encabezado de tipo de contenido de `application/octet-stream`, `image/*`, `video/*` o `audio/*`.
- La respuesta no se puede leer como una cadena válida. Por ejemplo, datos binarios o de imagen.

Patrones de integración de servicios

AWS Step Functions se integra con los servicios directamente en el idioma de los Estados de Amazon. Puede controlar estos servicios de AWS mediante tres patrones de integración de servicios:

- Llame a un servicio y deje que Step Functions avance al siguiente estado de manera inmediata después de que obtenga una respuesta HTTP.
- Llame a un servicio y haga que Step Functions espere a que finalice un trabajo.
- Llame a un servicio con un token de tarea y haga que Step Functions espere hasta que se devuelva dicho token con una carga.

Cada uno de estos patrones de integración de servicios se controla por cómo se crea un URI en el campo `"Resource"` de la definición de tarea.

Formas de llamar a un servicio integrado

- Respuesta de la solicitud
- Ejecutar un trabajo (.sync)
- Cómo esperar una devolución de llamada con el token de tarea

Para obtener información sobre la configuración AWS Identity and Access Management (IAM) de los servicios integrados, consultePolíticas de IAM para servicios integrados.

Respuesta de la solicitud

Cuando se especifica un servicio en la cadena `"Resource"` del estado de la tarea y solo se proporciona el recurso, Step Functions esperará una respuesta HTTP y, a continuación, avanzará al siguiente estado. Step Functions no esperará a que se complete un trabajo.

En el siguiente ejemplo se muestra cómo publicar un tema de Amazon SNS.

```
"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"
```

```
}
```

Este ejemplo hace referencia a la API Publish de Amazon SNS. El flujo de trabajo avanza hasta el siguiente estado después de llamar a la API `Publish`.

> **Tip**
>
> Para implementar un ejemplo de flujo de trabajo que utilice el patrón de integración del servicio Request Response Cuenta de AWS, consulte el módulo 2: Request Response of The AWS Step Functions Workshop.

Ejecutar un trabajo (.sync)

En el caso de los servicios integrados, como AWS Batch Amazon ECS, Step Functions puede esperar a que se complete una solicitud antes de pasar al siguiente estado. Para que Step Functions espere, especifique el campo `"Resource"` en la definición de estado de tarea con el sufijo `.sync` añadido después del URI del recurso.

Por ejemplo, al enviar un AWS Batch trabajo, utilice el `"Resource"` campo de la definición de la máquina de estados, tal y como se muestra en este ejemplo.

```
"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"
}
```

Tener la parte `.sync` añadida al nombre de recurso de Amazon (ARN) del recurso significa que Step Functions espera a que finalice el trabajo. Después de llamar a `submitJob` de AWS Batch , el flujo de trabajo se detiene. Cuando finaliza el trabajo, Step Functions avanza al siguiente estado. Para obtener más información, consulte el proyecto AWS Batch de muestra:Administración de un trabajo por lotes (AWS Batch, Amazon SNS).

Si se anula una tarea que utiliza este patrón de integración de servicios (`.sync`) y Step Functions no puede cancelarla, es posible que se incurra en cargos adicionales por el servicio integrado. Una tarea se puede anular si:

- La ejecución de la máquina de estado se detiene.
- Una ramificación diferente de un estado Parallel produce un error no detectado.
- La iteración de un estado de Map produce un error no detectado.

Step Functions hará todo lo posible por cancelar la tarea. Por ejemplo, si se anula una tarea `states:startExecution.sync` de Step Functions, se llamará a la acción de la API `StopExecution` de Step Functions. Sin embargo, es posible que Step Functions no pueda cancelar la tarea. Entre los motivos se incluyen, entre otros:

- Su rol de ejecución de IAM carece de permiso para realizar la llamada a la API correspondiente.
- Se produjo una interrupción temporal del servicio.

Cuando utiliza el patrón de integración de servicios `.sync`, Step Functions utiliza sondeos que consumen la cuota y los eventos asignados para supervisar el estado de un trabajo. Para `.sync` las invocaciones dentro de la misma cuenta, Step Functions usa EventBridge eventos y sondea las API que especifique en el `Task` estado. Para las invocaciones `.sync` entre cuentas, Step Functions solo utiliza sondeos. Por ejemplo, para`states:StartExecution.sync`, Step Functions realiza sondeos en la DescribeExecutionAPI y utiliza la cuota asignada.

> **Tip**
>
> Para implementar un ejemplo de flujo de trabajo que utilice el patrón de integración del servicio Run a Job (.sync) en su caso Cuenta de AWS, consulte el Módulo 3: Run a Job (.sync) de The AWS Step Functions Workshop.

Para ver una lista de los servicios integrados que admiten la espera para que finalice un trabajo (`.sync`), consulte Integraciones optimizadas para Step Functions.

> **Note**
>
> Las integraciones de servicios que utilizan el patrón `.sync` requieren permisos de IAM adicionales. Para obtener más información, consulte Políticas de IAM para servicios integrados.

En algunos casos, es posible que desee que Step Functions continúe con el flujo de trabajo antes de que el trabajo finalice por completo. Puede lograrlo de la misma manera que cuando utiliza el patrón de integración de servicios Cómo esperar una devolución de llamada con el token de tarea. Para ello, pase un token de tarea a su trabajo y, a continuación, devuélvalo mediante una llamada a la API `SendTaskSuccess` o `SendTaskFailure`. Step Functions utilizará los datos que proporcione en esa llamada para completar la tarea, dejar de supervisar el trabajo y continuar con el flujo de trabajo.

Cómo esperar una devolución de llamada con el token de tarea

Las tareas de devolución de llamada proporcionan una forma de detener un flujo de trabajo hasta que se devuelva un token de tarea. Una tarea podría tener que esperar la aprobación por parte de una persona, integrarse con un tercero o llamar a sistemas heredados. Para tareas como estas, puede pausar Step Functions hasta que la ejecución del flujo de trabajo alcance la cuota de servicio de un año (consulte Cuotas relacionadas con la limitación controlada de estados) y esperar a que se complete un proceso o flujo de trabajo externo. En estas situaciones, Step Functions te permite pasar un token de tarea a las integraciones de servicios del AWS SDK y también a algunas integraciones de servicios optimizadas. La tarea se detendrá hasta que se reciba dicho token de tarea de nuevo con una llamada `SendTaskSuccess` o `SendTaskFailure`.

Si se agota el tiempo de espera de un estado `Task` que utiliza el token de tarea de devolución de llamada, se genera un nuevo token aleatorio. Puede acceder a los tokens de tarea desde el objeto context.

> **Note**
>
> El token de tarea debe contener al menos un carácter y no puede superar los 1024 caracteres.

Para utilizarla `.waitForTaskToken` con una integración de AWS SDK, la API que utilices debe tener un campo de parámetros en el que colocar el token de tarea.

> **Note**
>
> Debes transferir los tokens de tareas de los directores de la misma AWS cuenta. Los tokens no funcionarán si los envías desde los directores de otra AWS cuenta.

> **Tip**
>
> Para implementar un ejemplo de flujo de trabajo que utilice un patrón de integración de servicios de token de tareas de devolución de llamadas Cuenta de AWS, consulte el módulo 4: Espere una devolución de llamada con el token de tareas de The Workshop. AWS Step Functions

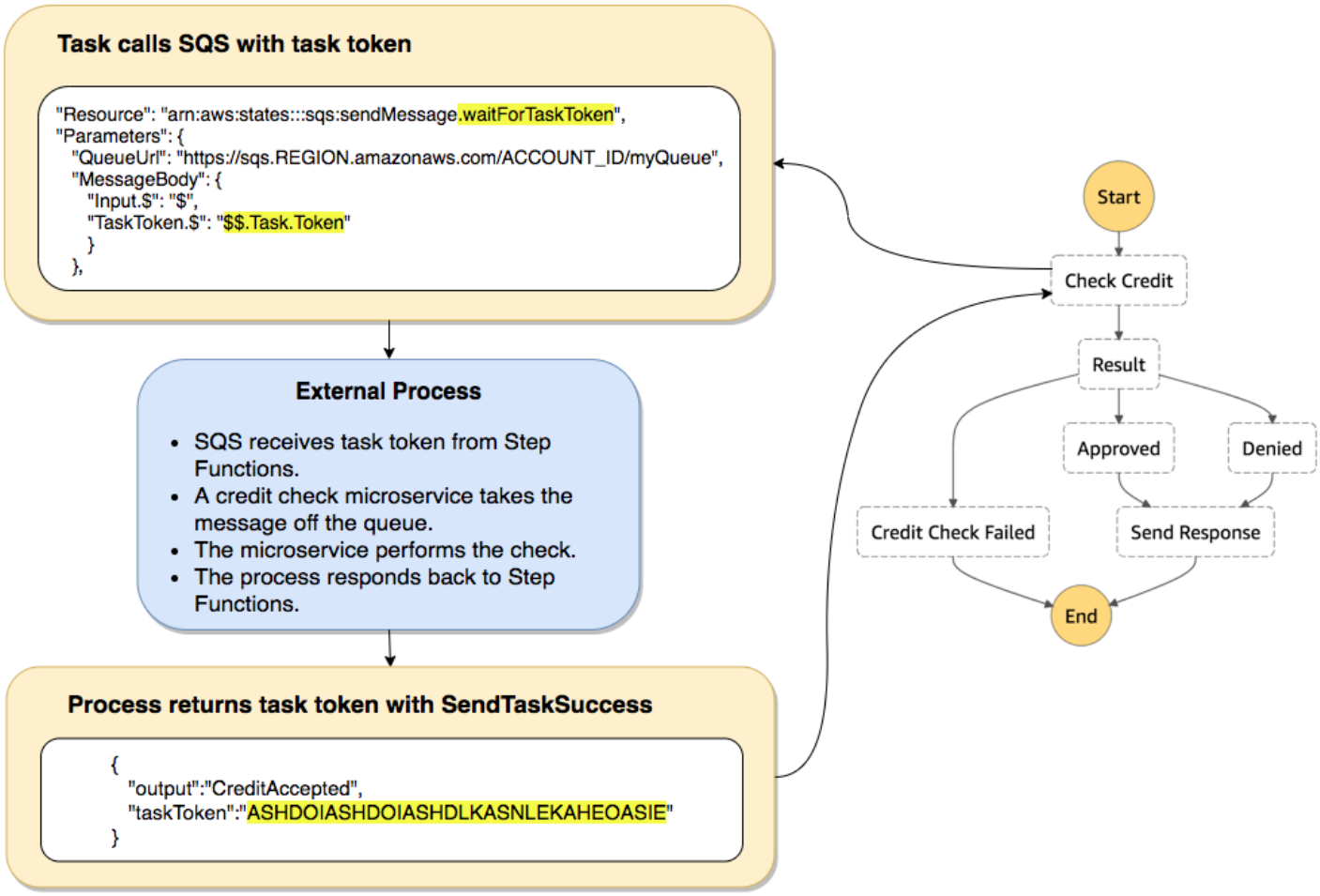
Para ver una lista de los servicios integrados que admiten la espera de un token de tarea (`.waitForTaskToken`), consulte Integraciones optimizadas para Step Functions.

Temas

- Ejemplo de token de tarea
- Cómo obtener un token del objeto context
- Cómo configurar un tiempo de espera de latido para una tarea en espera

Ejemplo de token de tarea

En este ejemplo, un flujo de trabajo de Step Functions debe integrarse con un microservicio externo para realizar una comprobación de crédito como parte de un flujo de trabajo de aprobación. Step Functions publica un mensaje de Amazon SQS que incluye un token de tarea como parte del mensaje. Un sistema externo se integra con Amazon SQS y obtiene el mensaje de la cola. Cuando dicho proceso haya terminado, devuelve el resultado y el token de tarea original. Posteriormente, Step Functions continúa con su flujo de trabajo.



El campo `"Resource"` de la definición de tarea que hace referencia a Amazon SQS incluye `.waitForTaskToken` añadido al final.

```
"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"
}
```

Este indica a Step Functions que se detenga y espere el token de tarea. Cuando se especifica un recurso con `.waitForTaskToken`, se puede acceder al token de tarea en el campo `"Parameters"` de la definición de estado con una designación de ruta especial (`$$.Task.Token`). La `$$.` inicial designa que la ruta accede al objeto context y obtiene el token de tarea para la tarea actual en una ejecución en curso.

Cuando finalice, el servicio externo llama a `SendTaskSuccess` o `SendTaskFailure` con el `taskToken` incluido. Solo entonces el flujo de trabajo avanza al siguiente estado.

> **Note**
>
> Para evitar la espera de forma indefinida si un proceso no envía el token de tarea con `SendTaskSuccess` o `SendTaskFailure`, consulte Cómo configurar un tiempo de espera de latido para una tarea en espera.

Cómo obtener un token del objeto context

El objeto context es un objeto JSON interno que contiene información acerca de la ejecución. Al igual que la entrada de estado, se puede acceder con una ruta desde el campo `"Parameters"` durante una ejecución. Al obtener acceso desde una definición de tarea, incluye información acerca de la ejecución específica, incluido el token de tarea.

```
{
    "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"
    }
}
```

Puede acceder al token de tarea mediante una ruta especial desde dentro del campo `"Parameters"` de la definición de tarea. Para acceder a la entrada o al objeto context, especifique primero que el parámetro será una ruta añadiendo un `.$` al nombre del parámetro. El siguiente ejemplo especifica los nodos de la entrada y del objeto context en una especificación de `"Parameters"`.

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

En ambos casos, la adición de `.$` al nombre del parámetro indica a Step Functions que espere una ruta. En el primer caso, `"$"` es una ruta que incluye toda la entrada. En el segundo caso, `$$.` especifica que la ruta accederá al objeto context y `$$.Task.Token` establece el parámetro en el valor del token de tarea en el objeto context de una ejecución en curso.

En el ejemplo de Amazon SQS, `.waitForTaskToken` en el campo `"Resource"` indica a Step Functions que espere a que se devuelva el token de tarea. El parámetro `"TaskToken.$": "$$.Task.Token"` pasa dicho token como parte del mensaje de Amazon SQS.

```
"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"
}
```

Para obtener más información acerca del objeto context, consulte Objeto Context (Contexto) en la sección Procesamiento de entrada y salida de esta guía.

Cómo configurar un tiempo de espera de latido para una tarea en espera

Una tarea que está a la espera de un token de tarea esperará hasta que la ejecución alcance la cuota de servicio de un año (consulte Cuotas relacionadas con la limitación controlada de estados). Para evitar que las ejecuciones se bloqueen, puede configurar un intervalo de tiempo de espera de latido en la definición de máquina de estado. Utilice el campo `HeartbeatSeconds` para especificar el intervalo de tiempo de espera.

```
{
  "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
    }
  }
}
```

En esta definición de máquina de estado, una tarea envía un mensaje a Amazon SQS y espera a que un proceso externo devuelva la llamada con el token de tarea proporcionado. El campo `"HeartbeatSeconds": 600` establece el intervalo de tiempo de espera de latido en 10 minutos. La tarea esperará a que el token de tarea se devuelva con una de estas acciones de la API:

- `SendTaskSuccess`
- `SendTaskFailure`
- `SendTaskHeartbeat`

Si la tarea en espera no recibe un token de tarea válido dentro de ese periodo de 10 minutos, se produce un error con el nombre `States.Timeout` en la tarea.

Para obtener más información, consulte el proyecto de ejemplo de devolución de llamada Ejemplo de patrón de devolución de llamadas (Amazon SQS, Amazon SNS, Lambda) .

Cómo pasar parámetros a una API de servicio

Utilice el campo `Parameters` de un estado `Task` para controlar qué parámetros se pasan a una API de servicio.

Dentro del campo `Parameters`, debe usar la forma plural de los parámetros de la matriz en una acción de API. Por ejemplo, si utiliza el campo Filtrar de la acción de API `DescribeSnapshots` para la integración con Amazon EC2, debe definir el campo como `Filters`. Si no se utiliza el plural, Step Functions devuelve el siguiente error:

```
The field Filter is not supported by Step Functions.
```

Cómo pasar JSON estático como parámetros

Puede incluir un objeto JSON directamente en la definición de la máquina de estado para pasárselo a un recurso como parámetro.

Por ejemplo, para establecer el parámetro `RetryStrategy` de la API `SubmitJob` para AWS Batch, podría incluir lo siguiente en los parámetros.

```
"RetryStrategy": {
  "attempts": 5
}
```

También puede pasar varios parámetros con JSON estático. Este es otro ejemplo más completo en el que se utilizan los campos `Resource` y `Parameters` para especificar una tarea que publica un tema de Amazon SNS llamado *`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"
        }
      }
  },
```

Cómo transferir la entrada de un estado como parámetros mediante rutas

Puede pasar partes de la entrada del estado como parámetros mediante el uso de rutas. Una ruta es una cadena que empieza por $, que puede utilizar para identificar componentes en texto JSON. Las rutas de Step Functions utilizan la sintaxis JsonPath.

Para especificar que un parámetro utilice una ruta, termine el nombre del parámetro con `.$`. Por ejemplo, si la entrada de estado contiene texto dentro de un nodo denominado`message`, puede pasar ese texto como parámetro mediante una ruta.

Supongamos que tenemos la siguiente entrada de estado:

```
{
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

Para pasar el valor del nodo denominado `message` como parámetro, especifique la siguiente sintaxis:

```
"Parameters": {"myMessage.$": "$.input.message"},
```

A continuación, Step Functions pasará el valor `foo` como parámetro.

Para obtener más información acerca del uso de parámetros en Step Functions, consulte los siguientes temas:

- Procesamiento de entrada y salida
- InputPath, Parámetros y ResultSelector

Cómo transferir nodos del objeto context como parámetros

Además del contenido estático y de los nodos de la entrada del estado, puede pasar nodos desde el objeto context como parámetros. El objeto context son datos JSON dinámicos presentes durante la ejecución de una máquina de estado. Incluye información acerca de la máquina de estado y la ejecución actual. Puede acceder al objeto context con una ruta en el campo `"Parameters"` de una definición de estado.

Para obtener más información acerca del objeto context y cómo acceder a los datos desde el campo `"Parameters"`, consulte los siguientes temas:

- Objeto Context (Contexto)
- Acceso al objeto de contexto
- Cómo obtener un token del objeto context

Registro de cambios para las integraciones AWS de SDK compatibles

La siguiente tabla resume cuándo los servicios se integraron inicialmente con Step Functions y cuándo se actualizó su API de integración por última vez. Para obtener más información sobre el uso de las integraciones, consulte. AWS Integraciones de servicios SDK

> **Important**
>
> El soporte para las acciones de la API se publica trimestralmente. Es posible que las actualizaciones de las acciones ya compatibles, como los nuevos parámetros, no estén disponibles de forma inmediata.

Servicio	Soporte inicial	Actualizado	
AWS AppFabric	18 de enero de 2024		
B2B Data Interchange	18 de enero de 2024		

Servicio	Soporte inicial	Actualizado	
Exportaciones de datos de AWS	18 de enero de 2024		
Amazon Bedrock	18 de enero de 2024		
Amazon Bedrock Agents	18 de enero de 2024		
Amazon Bedrock Runtime Agents	18 de enero de 2024		
Amazon Bedrock Runtime	18 de enero de 2024		
Amazon CloudFront KeyValueStore	18 de enero de 2024		
Amazon CodeGuru Security	18 de enero de 2024		
Centro de optimización de costes de AWS	18 de enero de 2024		
Amazon DataZone	18 de enero de 2024		
Amazon EKS Auth	18 de enero de 2024		
AWS Entity Resolution	18 de enero de 2024		
capa gratuita de AWS	18 de enero de 2024		
Amazon Inspector Scan	18 de enero de 2024		
AWS Launch Wizard	18 de enero de 2024		

Servicio	Soporte inicial	Actualizado	
Amazon Managed Blockchain Query	18 de enero de 2024		
AWS Elemental MediaPackage V2	18 de enero de 2024		
AWS HealthImaging	18 de enero de 2024		
Network Manager	18 de enero de 2024		
AWS Payment Cryptography	18 de enero de 2024		
AWS Payment Cryptography Data	18 de enero de 2024		
AWS Private CA Connector for Active Directory	18 de enero de 2024		
Amazon Q Business	18 de enero de 2024		
Amazon Q Connect	18 de enero de 2024		
AWS re:Post	18 de enero de 2024		
Amazon Timestream Query	18 de enero de 2024		
Amazon Timestream Write	18 de enero de 2024		
Trusted Advisor	18 de enero de 2024		
Verified Permissions	18 de enero de 2024		
Amazon WorkSpaces Thin Client	18 de enero de 2024		

Servicio	Soporte inicial	Actualizado	
AWS CloudTrail Data	16 de junio de 2023		
Amazon CloudWatch Internet Monitor	16 de junio de 2023		
Amazon Interacti ve Video Service RealTime	16 de junio de 2023		
AWS IoT TwinMaker	16 de junio de 2023		
Amazon OpenSearch Ingestion	16 de junio de 2023		
AWS Telco Network Builder	16 de junio de 2023		
Amazon VPC Lattice	16 de junio de 2023		
AWS Backup Storage	17 de febrero de 2023		
Amazon Chime Media Pipelines	17 de febrero de 2023		
Amazon Chime Voice	17 de febrero de 2023		
AWS Clean Rooms	17 de febrero de 2023	18 de enero de 2024	
Amazon CodeCatalyst	17 de febrero de 2023		
Amazon Connect Cases	17 de febrero de 2023		
AWS Control Tower	17 de febrero de 2023		
Amazon DocumentDB Elastic Clusters	17 de febrero de 2023		

Servicio	Soporte inicial	Actualizado	
Amazon EMR Serverless	17 de febrero de 2023		
Amazon IVS Chat	17 de febrero de 2023		
Amazon Kendra Intelligent Ranking	17 de febrero de 2023		
AWS HealthOmics	17 de febrero de 2023		
Amazon Redshift Serverless	17 de febrero de 2023		
Amazon Security Lake	17 de febrero de 2023		
AWS Health	17 de febrero de 2023		
AWS IoT FleetWise	17 de febrero de 2023		
AWS IoT RoboRunner	17 de febrero de 2023		
AWS Mainframe Modernization	17 de febrero de 2023		
Orquestador de AWS Migration Hub	17 de febrero de 2023		
AWS Private 5G	17 de febrero de 2023		
Explorador de recursos de AWS	17 de febrero de 2023		
AWS SimSpace Weaver	17 de febrero de 2023		
AWS Support App	17 de febrero de 2023		

Servicio	Soporte inicial	Actualizado	
CloudWatch Observability Access Manager	17 de febrero de 2023		
EventBridge Pipes	17 de febrero de 2023		
EventBridge Scheduler	17 de febrero de 2023		
IAM Roles Anywhere	17 de febrero de 2023		
Kinesis Video WebRTC Storage	17 de febrero de 2023		
License Manager Linux Subscriptions	17 de febrero de 2023		
License Manager User Subscriptions	17 de febrero de 2023		
OpenSearch Serverless	17 de febrero de 2023		
Route 53 ARC Zonal Shift	17 de febrero de 2023		
SageMaker Geospatial	17 de febrero de 2023		
SageMaker Metrics	17 de febrero de 2023		
Systems Manager for SAP	17 de febrero de 2023		
AWS Account Management	19 de abril de 2022		

Servicio	Soporte inicial	Actualizado	
AWS Amplify	30 de septiembre de 2021		
AWS App Mesh	30 de septiembre de 2021		
AWS App Runner	30 de septiembre de 2021	17 de febrero de 2023	
AWS AppConfig	30 de septiembre de 2021		
AWS AppConfig Data	19 de abril de 2022		
AWS AppSync	30 de septiembre de 2021	17 de febrero de 2023	
AWS Application Discovery Service	30 de septiembre de 2021		
AWS Application Migration Service	30 de septiembre de 2021		
AWS Audit Manager	30 de septiembre de 2021		
AWS Auto Scaling Plans	30 de septiembre de 2021		
AWS Backup	30 de septiembre de 2021	17 de febrero de 2023	
AWS Backup gateway	19 de abril de 2022	17 de febrero de 2023	
AWS Batch	30 de septiembre de 2021	17 de febrero de 2023	

Servicio	Soporte inicial	Actualizado	
AWS Billing Conductor	26 de julio de 2022	17 de febrero de 2023	
AWS Budgets	30 de septiembre de 2021		
AWS Certificate Manager	30 de septiembre de 2021		
AWS Private Certificate Authority	30 de septiembre de 2021		
AWS Cloud Map	30 de septiembre de 2021		
AWS Cloud9	30 de septiembre de 2021		
AWS CloudFormation	30 de septiembre de 2021	17 de febrero de 2023	
AWS CloudHSM	30 de septiembre de 2021		
AWS CloudHSM	30 de septiembre de 2021		
AWS CloudTrail	30 de septiembre de 2021	17 de febrero de 2023	
AWS Cloud Control	19 de abril de 2022		
AWS CodeBuild	30 de septiembre de 2021		
AWS CodeCommit	30 de septiembre de 2021	17 de febrero de 2023	

Servicio	Soporte inicial	Actualizado	
AWS CodeDeploy	30 de septiembre de 2021		
AWS CodePipeline	30 de septiembre de 2021		
AWS CodeStar	30 de septiembre de 2021		
AWS CodeStar	30 de septiembre de 2021		
AWS CodeStar	30 de septiembre de 2021		
AWS Compute Optimizer	30 de septiembre de 2021	17 de febrero de 2023	
AWS Config	30 de septiembre de 2021	26 de julio de 2022	
AWS Cost Explorer Service	30 de septiembre de 2021	17 de febrero de 2023	
AWS Cost and Usage Report	30 de septiembre de 2021		
AWS Data Exchange	30 de septiembre de 2021	26 de julio de 2022	
AWS Data Pipeline	30 de septiembre de 2021		
AWS DataSync	30 de septiembre de 2021	26 de julio de 2022	
AWS Database Migration Service	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
AWS Device Farm	30 de septiembre de 2021		
AWS Direct Connect	30 de septiembre de 2021		
AWS Directory Service	30 de septiembre de 2021	17 de febrero de 2023	
AWS EC2 Instance Connect	30 de septiembre de 2021		
AWS Elastic Beanstalk	30 de septiembre de 2021		
AWS Elemental MediaLive	30 de septiembre de 2021		
AWS Elemental MediaPackage	30 de septiembre de 2021		
AWS Elemental MediaPackage VOD	30 de septiembre de 2021		
AWS Elemental MediaStore	30 de septiembre de 2021		
AWS Fault Injection Service	30 de septiembre de 2021		
AWS Firewall Manager	30 de septiembre de 2021	17 de febrero de 2023	
AWS Glue	30 de septiembre de 2021	17 de febrero de 2023	
AWS Glue DataBrew	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
AWS IoT Greengrass	30 de septiembre de 2021		
AWS Ground Station	30 de septiembre de 2021	17 de febrero de 2023	
AWS Identity and Access Management	30 de septiembre de 2021		
AWS IoT	30 de septiembre de 2021	17 de febrero de 2023	
AWS IoT 1-Click	30 de septiembre de 2021		
AWS IoT Analytics	30 de septiembre de 2021		
AWS IoT Core Device Advisor	30 de septiembre de 2021		
AWS IoT Events	30 de septiembre de 2021		
Datos de AWS IoT Events	30 de septiembre de 2021		
AWS IoT Fleet Hub	30 de septiembre de 2021		
AWS IoT Greengrass Version 2	30 de septiembre de 2021		
AWS IoT Jobs Data Plane	30 de septiembre de 2021		
AWS IoT Secure Tunneling	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
AWS IoT SiteWise	30 de septiembre de 2021	17 de febrero de 2023	
AWS IoT Things Graph	30 de septiembre de 2021		
AWS IoT Wireless	30 de septiembre de 2021	17 de febrero de 2023	
AWS Key Management Service	30 de septiembre de 2021	26 de julio de 2022	
AWS Lake Formation	30 de septiembre de 2021	17 de febrero de 2023	
AWS Lambda	30 de septiembre de 2021	17 de febrero de 2023	
AWS License Manager	30 de septiembre de 2021	17 de febrero de 2023	
AWS Marketplace	30 de septiembre de 2021	17 de febrero de 2023	
AWS Marketplace Commerce Analytics	30 de septiembre de 2021		
AWS Marketplace Entitlement Service	30 de septiembre de 2021		
AWS Elemental MediaTailor	30 de septiembre de 2021	26 de julio de 2022	
AWS Migration Hub	30 de septiembre de 2021		
AWS Migration Hub Config	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Recomendaciones de estrategias de AWS Migration Hub	19 de abril de 2022	17 de febrero de 2023	
AWS Mobile	30 de septiembre de 2021		
AWS Network Firewall	30 de septiembre de 2021		
AWS OpsWorks	30 de septiembre de 2021		
AWS OpsWorks CM	30 de septiembre de 2021		
AWS Organizations	30 de septiembre de 2021	17 de febrero de 2023	
AWS Outposts	30 de septiembre de 2021		
AWS Panorama	19 de abril de 2022	17 de febrero de 2023	
Amazon Relational Database Service Performance Insights	30 de septiembre de 2021		
Lista de precios de AWS	30 de septiembre de 2021		
Amazon Relational Database Service	30 de septiembre de 2021		
AWS Resilience Hub	19 de abril de 2022		
AWS Resource Access Manager	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
AWS Resource Groups	30 de septiembre de 2021	17 de febrero de 2023	
AWS Resource Groups Tagging API	30 de septiembre de 2021		
AWS RoboMaker	30 de septiembre de 2021		
AWS IAM Identity Center	30 de septiembre de 2021	17 de febrero de 2023	
AWS SSO OIDC	30 de septiembre de 2021		
AWS Secrets Manager	30 de septiembre de 2021		
AWS Security Token Service	30 de septiembre de 2021		
AWS Security Hub	30 de septiembre de 2021		
AWS Server Migration Service	30 de septiembre de 2021		
AWS Service Catalog	30 de septiembre de 2021		
AWS Service Catalog AppRegistry	30 de septiembre de 2021	17 de febrero de 2023	
AWS Shield	30 de septiembre de 2021		
AWS Signer	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
AWS IAM Identity Center	30 de septiembre de 2021		
AWS IAM Identity Center Admin	30 de septiembre de 2021		
AWS Step Functions	30 de septiembre de 2021	17 de febrero de 2023	
AWS Storage Gateway	30 de septiembre de 2021		
AWS Support	30 de septiembre de 2021		
AWS Transfer Family	30 de septiembre de 2021	17 de febrero de 2023	
AWS WAF	30 de septiembre de 2021		
AWS WAF Regional	30 de septiembre de 2021		
AWS WAFV2	30 de septiembre de 2021		
AWS Well-Architected Tool	30 de septiembre de 2021	17 de febrero de 2023	
AWS X-Ray	30 de septiembre de 2021	17 de febrero de 2023	
AWS Marketplace Metering Service	30 de septiembre de 2021		
AWS Serverless Application Repository	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
AWS Identity and Access Management Access Analyzer	30 de septiembre de 2021		
Alexa for Business	30 de septiembre de 2021		
Amazon API Gateway	30 de septiembre de 2021	17 de febrero de 2023	
Amazon API Gateway	30 de septiembre de 2021		
Amazon AppIntegrations	30 de septiembre de 2021		
Amazon AppStream 2.0	30 de septiembre de 2021		
Amazon AppFlow	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Athena	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Augmented AI	30 de septiembre de 2021		
Amazon Braket	30 de septiembre de 2021		
Amazon Chime	30 de septiembre de 2021		
Amazon Chime Meetings	19 de abril de 2022	17 de febrero de 2023	

Servicio	Soporte inicial	Actualizado	
Amazon Cloud Directory	30 de septiembre de 2021		
Amazon CloudFront	30 de septiembre de 2021	17 de febrero de 2023	
Amazon CloudSearch	30 de septiembre de 2021		
Amazon CloudWatch	30 de septiembre de 2021	17 de febrero de 2023	
Amazon CloudWatch Application Insights	30 de septiembre de 2021		
CloudWatch Evidently	19 de abril de 2022		
Amazon CloudWatch Logs	30 de septiembre de 2021		
Amazon CloudWatch RUM	19 de abril de 2022	17 de febrero de 2023	
Amazon CloudWatch Synthetics	30 de septiembre de 2021		
Amazon CodeGuru Profiler	30 de septiembre de 2021		
Amazon CodeGuru Reviewer	30 de septiembre de 2021		
Amazon Cognito	30 de septiembre de 2021		
Amazon Cognito Identity Provider	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon Cognito Sync	30 de septiembre de 2021		
Amazon Comprehend	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Comprehend Medical	30 de septiembre de 2021		
Amazon Connect Contact Lens	30 de septiembre de 2021		
Amazon Connect Participant Service	30 de septiembre de 2021		
Amazon Connect	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Connect Voice ID	19 de abril de 2022		
Amazon Connect Wisdom	19 de abril de 2022		
Amazon Data Lifecycle Manager	30 de septiembre de 2021		
Amazon Detective	30 de septiembre de 2021		
Amazon DevOps Guru	30 de septiembre de 2021	26 de julio de 2022	
Amazon DocumentDB (with MongoDB compatibility)	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon DynamoDB	30 de septiembre de 2021	17 de febrero de 2023	
Amazon DynamoDB Streams	30 de septiembre de 2021		
Amazon EC2 Container Registry	30 de septiembre de 2021		
Amazon EC2 Container Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon EC2 Systems Manager	30 de septiembre de 2021	17 de febrero de 2023	
Amazon EMR	30 de septiembre de 2021	17 de febrero de 2023	
Amazon ElastiCache	30 de septiembre de 2021		
Amazon Elastic Inference	30 de septiembre de 2021		
Amazon Elastic Block Store	30 de septiembre de 2021		
Amazon Elastic Compute Cloud	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Elastic Container Registry Public	30 de septiembre de 2021		
Amazon Elastic File System	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon Elastic Kubernetes Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon EMR	30 de septiembre de 2021		
Amazon Elastic Transcoder	30 de septiembre de 2021		
Amazon OpenSearch Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon OpenSearch Service	19 de abril de 2022	17 de febrero de 2023	
Amazon EventBridge	30 de septiembre de 2021	17 de febrero de 2023	
Amazon FSx	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Forecast Query	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Forecast Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Fraud Detector	30 de septiembre de 2021		
Amazon GameLift	30 de septiembre de 2021	17 de febrero de 2023	
Amazon GameSparks	26 de julio de 2022		
Amazon S3 Glacier	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon GuardDuty	30 de septiembre de 2021		
AWS HealthLake	30 de septiembre de 2021		
Amazon Honeycode	30 de septiembre de 2021		
Amazon Inspector	30 de septiembre de 2021		
Amazon Inspector V2	19 de abril de 2022		
Amazon Interactive Video Service	30 de septiembre de 2021		
Amazon Kendra	30 de septiembre de 2021		
Amazon Kinesis	30 de septiembre de 2021		
Amazon Kinesis Analytics	30 de septiembre de 2021		
Amazon Kinesis Analytics V2	30 de septiembre de 2021		
Amazon Kinesis Firehose	30 de septiembre de 2021		
Amazon Kinesis Video Signaling Channels	30 de septiembre de 2021		
Amazon Kinesis Video Streams	30 de septiembre de 2021	17 de febrero de 2023	

Servicio	Soporte inicial	Actualizado	
Amazon Kinesis Video Streams Archived Media	30 de septiembre de 2021		
Amazon Kinesis video stream	30 de septiembre de 2021		
Amazon Lex Model Building Service	30 de septiembre de 2021		
Amazon Lex Model Building Service V2	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Lex	30 de septiembre de 2021		
Amazon Lex Runtime V2	30 de septiembre de 2021		
Amazon Lightsail	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Location Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Lookout for Equipment	30 de septiembre de 2021		
Amazon Lookout for Metrics	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Lookout for Vision	30 de septiembre de 2021		
Amazon MQ	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon Macie	30 de septiembre de 2021		
Amazon Macie 2	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Managed Blockchain	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Managed Grafana	19 de abril de 2022	17 de febrero de 2023	
Amazon Managed Service for Prometheus	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Managed Streaming for Apache Kafka	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Managed Streaming for Apache Kafkax	19 de abril de 2022		
Amazon Managed Workflows for Apache Airflow	30 de septiembre de 2021		
Amazon Mechanical Turk	30 de septiembre de 2021		
Amazon MemoryDB for Redis	19 de abril de 2022	17 de febrero de 2023	
Amazon Nimble Studio	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon Personalize	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Personalize Events	30 de septiembre de 2021		
Amazon Personalize Runtime	30 de septiembre de 2021		
Amazon Pinpoint	30 de septiembre de 2021		
Amazon Pinpoint Email Service	30 de septiembre de 2021		
Amazon Pinpoint SMS and Voice Service	30 de septiembre de 2021		
Amazon Pinpoint SMS and Voice V2 Service	26 de julio de 2022		
Amazon Polly	30 de septiembre de 2021		
Amazon QLDB	30 de septiembre de 2021		
Amazon QLDB Session	30 de septiembre de 2021		
Amazon QuickSight	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Redshift	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon Redshift Data API	30 de septiembre de 2021		
Amazon Rekognition	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Relational Database Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Route 53	30 de septiembre de 2021		
Amazon Route 53 Recovery Control Config	30 de septiembre de 2021	26 de julio de 2022	
Amazon Route 53 Domains	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Route 53 Resolver	30 de septiembre de 2021		
Amazon S3 on Outposts	30 de septiembre de 2021	26 de julio de 2022	
Amazon SageMaker Runtime Feature Store Runtime	30 de septiembre de 2021		
Amazon SageMaker Runtime Runtime	30 de septiembre de 2021		
Amazon SageMaker	30 de septiembre de 2021	17 de febrero de 2023	
Amazon SageMaker Edge Manager	30 de septiembre de 2021		

Servicio	Soporte inicial	Actualizado	
Amazon Simple Email Service	30 de septiembre de 2021		
Amazon Simple Email Service V2	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Simple Notification Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Simple Queue Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Simple Storage Service	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Simple Workflow Service	30 de septiembre de 2021		
Amazon Textract	30 de septiembre de 2021	17 de febrero de 2023	
Amazon Transcribe	30 de septiembre de 2021		
Amazon Translate	30 de septiembre de 2021	17 de febrero de 2023	
Amazon WorkDocs	30 de septiembre de 2021	17 de febrero de 2023	
Amazon WorkMail	30 de septiembre de 2021	17 de febrero de 2023	
Amazon WorkMail Message Flow	30 de septiembre de 2021		
Amazon WorkSpaces	30 de septiembre de 2021	17 de febrero de 2023	

Servicio	Soporte inicial	Actualizado	
Amazon WorkSpaces Web	19 de abril de 2022	17 de febrero de 2023	
Amplify	30 de septiembre de 2021		
Amplify UI Builder	19 de abril de 2022	17 de febrero de 2023	
Application Auto Scaling	30 de septiembre de 2021		
Amazon EC2 Auto Scaling	30 de septiembre de 2021	17 de febrero de 2023	
CodeArtifact	30 de septiembre de 2021		
DynamoDB Accelerator	30 de septiembre de 2021		
EC2 Image Builder	30 de septiembre de 2021		
AWS Elastic Disaster Recovery	19 de abril de 2022	17 de febrero de 2023	
Elastic Load Balancing	30 de septiembre de 2021		
Elastic Load Balancing V2	30 de septiembre de 2021		
MediaConnect	30 de septiembre de 2021		
Amazon S3 Control	30 de septiembre de 2021	17 de febrero de 2023	

Servicio	Soporte inicial	Actualizado	
Recycle Bin for Amazon EBS	19 de abril de 2022	17 de febrero de 2023	
Savings Plans	30 de septiembre de 2021		
Amazon EventBridge Schema Registry	30 de septiembre de 2021		
Service Quotas	30 de septiembre de 2021		
AWS Snowball	30 de septiembre de 2021		

Proyectos de muestra para Step Functions

En la consola de AWS Step Functions, puede elegir una de las siguientes plantillas de inicio para implementar máquinas de estado en su Cuentas de AWS. Estas plantillas iniciales son proyectos de muestra listos para ejecutarse que crean automáticamente el prototipo y la definición del flujo de trabajo, así como todos los recursos de AWS relacionados con el proyecto.

Puede usar estos proyectos de muestra para implementarlos y ejecutarlos tal cual, o bien usar los prototipos de flujo de trabajo para desarrollarlas a partir de ellos. Si se basa en estos proyectos, Step Functions crea el prototipo del flujo de trabajo, pero no implementa los recursos que figuran en la definición del flujo de trabajo.

Cuando implementa los proyectos de muestra, se aprovisiona una máquina de estado plenamente funcional y se crean los recursos relacionados para que se ejecute. Al crear un proyecto de ejemplo, Step Functions utiliza AWS CloudFormation para crear los recursos relacionados a los que hace referencia la máquina de estado.

Temas

- Administración de un trabajo por lotes (AWS Batch, Amazon SNS)
- Administración de una tarea de contenedor (Amazon ECS, Amazon SNS)
- Transferencia de registros de datos (Lambda, DynamoDB y Amazon SQS)
- Encuesta sobre el estado del trabajo (Lambda,) AWS Batch
- Temporizador de tareas (Lambda, Amazon SNS)
- Ejemplo de patrón de devolución de llamadas (Amazon SQS, Amazon SNS, Lambda)
- Administrar un trabajo de Amazon EMR
- Ejecutar un trabajo de EMR Serverless
- Iniciar un flujo de trabajo dentro de un flujo de trabajo (Step Functions, Lambda)
- Procesamiento dinámico de datos con un estado Map
- Procesar un archivo CSV con Distributed Map
- Procesar datos de un bucket de Amazon S3 con Distributed Map
- Entrenamiento de un modelo de aprendizaje automático
- Ajuste de un modelo de aprendizaje automático
- Procesar mensajes de un volumen elevado desde Amazon SQS (flujos de trabajo rápidos)

- Ejemplo de punto de comprobación selectivo (flujos de trabajo rápidos)
- Creación de un AWS CodeBuild proyecto (CodeBuild, Amazon SNS)
- Preprocesamiento de datos y entrenamiento de un modelo de machine learning
- Ejemplos de orquestación de Lambda
- Iniciar una consulta de Athena
- Ejecutar varias consultas (Amazon Athena, Amazon SNS)
- Consulte conjuntos de datos de gran tamaño (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)
- Mantener los datos actualizados (Amazon Athena, Amazon S3,) AWS Glue
- Administración de un clúster de Amazon EKS
- Hacer una llamada a API Gateway
- Llamar a un microservicio que se ejecute en Fargate mediante la integración de API Gateway
- Envía un evento personalizado a EventBridge
- Invocar flujos de trabajo rápidos sincrónicos
- Ejecutar flujos de trabajo de ETL/ELT con Amazon Redshift (Lambda, API de datos de Amazon Redshift)
- Usar Step Functions y AWS Batch con control de errores
- Haz un abanico de AWS Batch trabajo
- AWS Batch con Lambda
- Encadenamiento de mensajes de IA con Amazon Bedrock

Administración de un trabajo por lotes (AWS Batch, Amazon SNS)

En este proyecto de muestra, se explica cómo enviar un trabajo de AWS Batch y, a continuación, una notificación de Amazon SNS que depende de si el trabajo se realiza correcta o incorrectamente. La implementación de este proyecto de muestra crea una máquina de estado de AWS Step Functions, un trabajo AWS Batch y un tema de Amazon SNS.

En este proyecto, Step Functions utiliza una máquina de estado para llamar al trabajo de AWS Batch de forma síncrona. A continuación, espera a que el trabajo se realice correcta o incorrectamente y envía un tema de Amazon SNS con un mensaje que informa sobre si el trabajo ha finalizado correctamente o con errores.

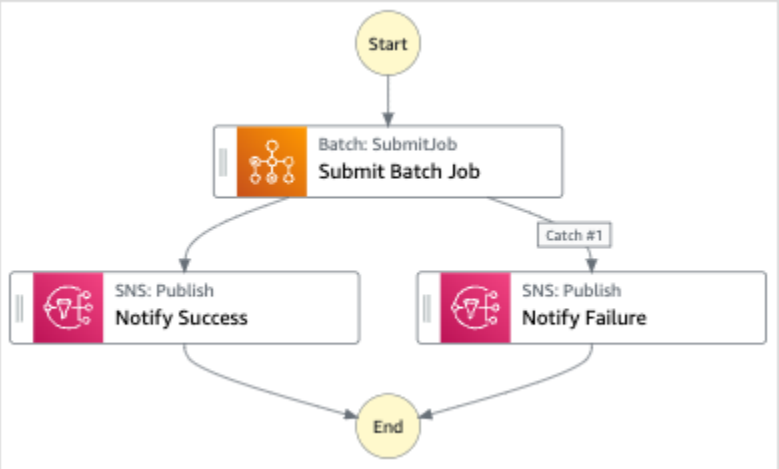
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Manage a batch job`** en el cuadro de búsqueda y, a continuación, seleccione Administración de un trabajo por lotes en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - ¿Un AWS Batch trabajo
   - Un tema de Amazon SNS
   - ¿Una máquina AWS Step Functions estatal
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Administración de un trabajo por lotes:

5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

  > **Important**
  >
  > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con AWS Batch Amazon SNS al pasar los parámetros directamente a esos recursos.

Explore este ejemplo de máquina de estados para ver cómo Step Functions controla AWS Batch Amazon SNS conectándose al nombre de recurso de Amazon (ARN) en el `Resource` campo y pasándolo `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
  "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
    }
  }
}
```

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Administración de una tarea de contenedor (Amazon ECS, Amazon SNS)

En este proyecto de muestra, se explica cómo ejecutar una tarea de AWS Fargate y, a continuación, enviar una notificación de Amazon SNS que depende de si el trabajo se realiza correcta o incorrectamente. La implementación de este proyecto de muestra creará una máquina de estado de AWS Step Functions, un clúster de Fargate y un tema de Amazon SNS.

En este proyecto, Step Functions utiliza una máquina de estado para llamar a la tarea de Fargate de forma síncrona. A continuación, espera a que la tarea se realice correcta o incorrectamente y envía un tema de Amazon SNS con un mensaje que informa sobre si el trabajo ha finalizado correctamente o con errores.

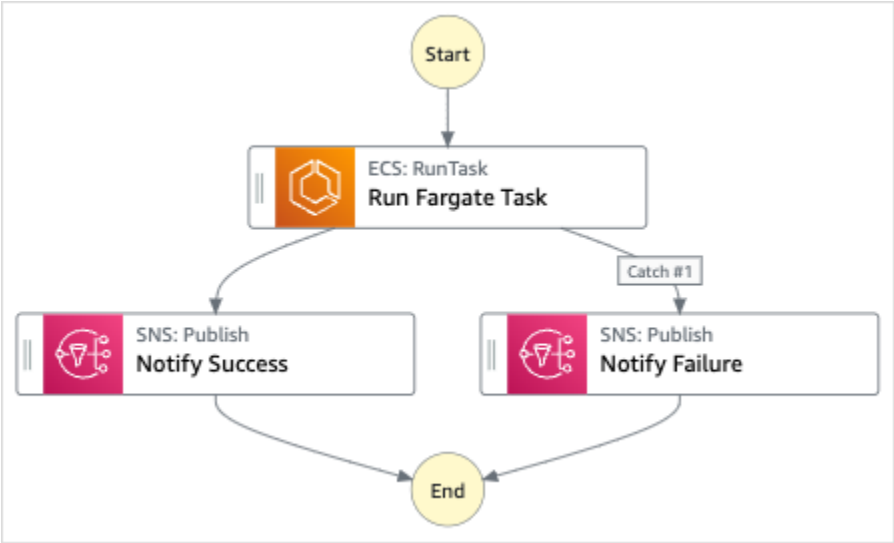
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Manage a container task`** en el cuadro de búsqueda y, a continuación, seleccione Administración de una tarea de contenedor en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - ¿Un AWS Fargate clúster
   - Un tema de Amazon SNS
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Administración de una tarea de contenedor:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

> **Tip**
>
> Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:
   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que

puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con AWS Fargate Amazon SNS al pasar los parámetros directamente a esos recursos. Desplácese hasta la máquina de estado de este ejemplo para ver cómo Step Functions utiliza una máquina de estado para llamar a la tarea de Fargate de forma sincrónica, espera a que la tarea se realice correctamente o produzca un error y envía un tema de Amazon SNS con un mensaje acerca de si el trabajo ha finalizado correctamente o ha producido un error.

Para obtener más información acerca de cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Es una práctica recomendada incluir solo los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Transferencia de registros de datos (Lambda, DynamoDB y Amazon SQS)

En este proyecto de muestra se ilustra cómo leer de forma iterativa los elementos de una tabla de Amazon DynamoDB y enviarlos a una cola de Amazon SQS mediante una máquina de estado de Step Functions. La implementación de este proyecto de muestra creará una máquina de estado de Step Functions, una tabla de DynamoDB, una función de AWS Lambda y una cola de Amazon SQS.

En este proyecto, Step Functions utiliza la función de Lambda para rellenar la tabla DynamoDB. La máquina de estado también usa un bucle de `for` para leer cada una de las entradas y, a continuación, envía cada entrada a una cola de Amazon SQS.

Paso 1: Crear la máquina de estado y aprovisionar recursos

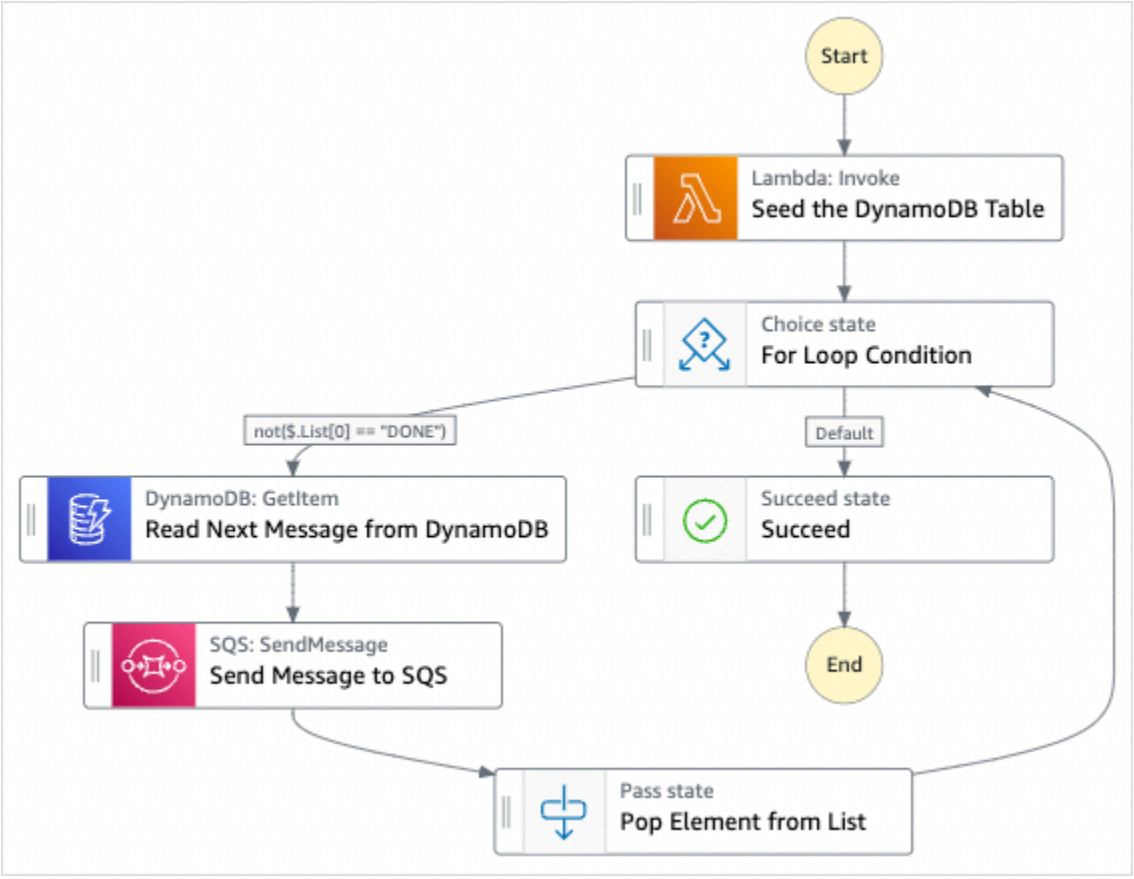
1. Abra la [consola de Step Functions](#) y seleccione Crear máquina de estado.

2. Escriba **`Transfer data records`** en el cuadro de búsqueda y, a continuación, seleccione Transferencia de registros de datos en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una función de Lambda para rellenar la tabla de DynamoDB
   - Una cola de Amazon SQS
   - Una tabla de DynamoDB
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Transferencia de registros de datos:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con DynamoDB y Amazon SQS pasando parámetros directamente a esos recursos.

Examine este ejemplo de máquina de estado para ver cómo Step Functions controla DynamoDB y Amazon SQS conectándose al Nombre de recurso de Amazon (ARN) del campo `Resource` y pasando `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

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

Para obtener más información sobre cómo pasar parámetros y administrar los resultados, consulte:

- Cómo pasar parámetros a una API de servicio
- ResultPath

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los

recursos relacionados. Es una práctica recomendada incluir solo los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Encuesta sobre el estado del trabajo (Lambda,) AWS Batch

Este proyecto de ejemplo crea una encuesta de AWS Batch empleos. Implementa una máquina de AWS Step Functions estados que se utiliza AWS Lambda para crear un bucle de `Wait` estados que comprueba un AWS Batch trabajo.

Este proyecto de ejemplo crea y configura todos los recursos para que el flujo de trabajo de Step Functions envíe un AWS Batch trabajo y espere a que ese trabajo se complete antes de finalizar correctamente.

> **Note**
>
> También puede implementar este patrón sin utilizar una función de Lambda. Para obtener información sobre cómo controlar AWS Batch directamente, consulteUso AWS Step Functions con otros servicios.

Este proyecto de ejemplo crea la máquina de estados, dos funciones Lambda y una AWS Batch cola, y configura los permisos de IAM relacionados.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

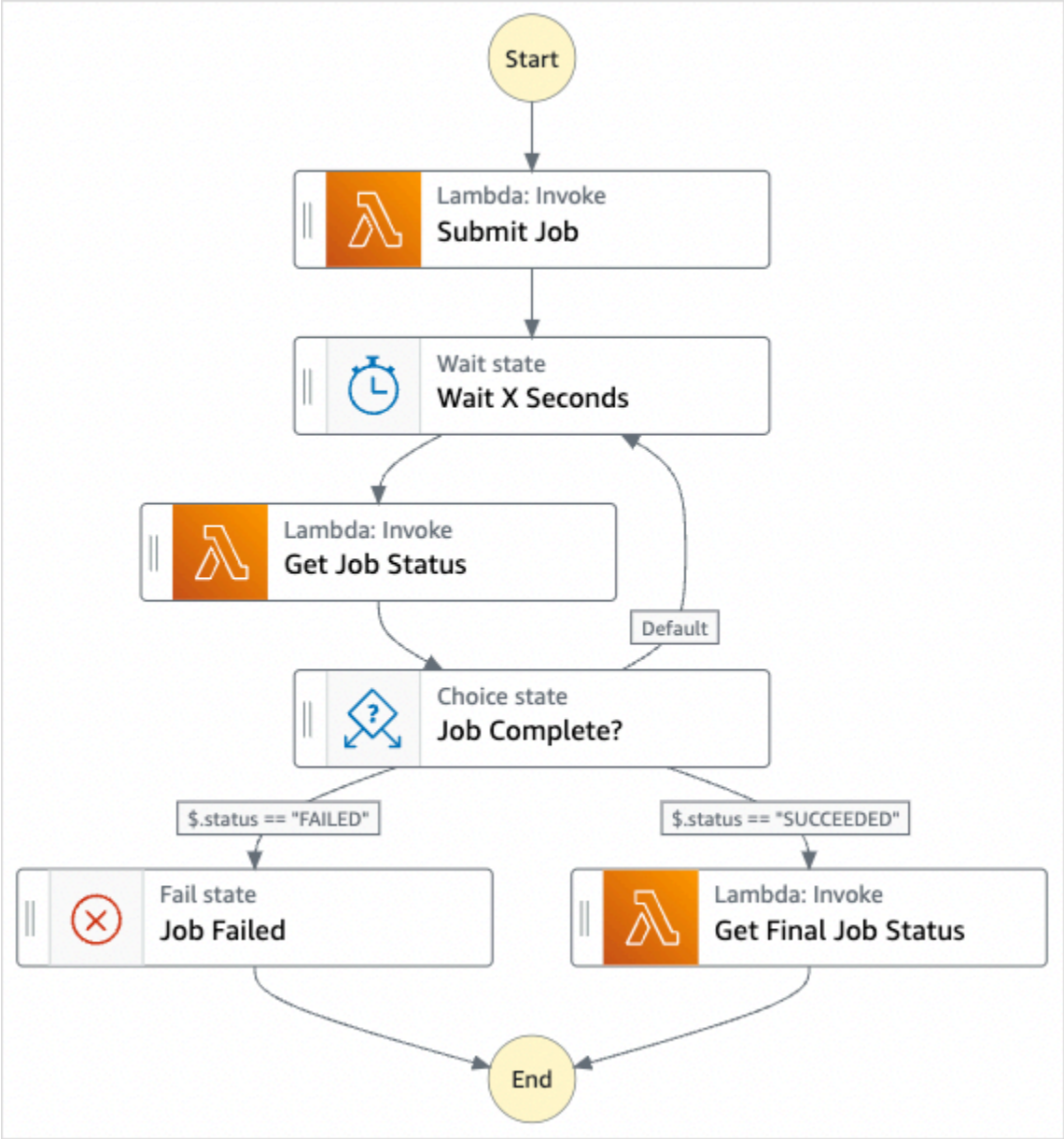
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Job Poller`** en el cuadro de búsqueda y, a continuación, seleccione Sondeador de un trabajo en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

- Tres funciones Lambda para enviar un AWS Batch trabajo, obtener el estado actual del AWS Batch trabajo enviado y el estado final del trabajo completado.
- ¿Un trabajo AWS Batch
- ¿Una máquina AWS Step Functions estatal
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Sondeador de un trabajo:



5. Elija Utilizar plantilla para continuar con la selección.

6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

     > **Tip**
     >
     > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

     Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

     El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

     Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

Una vez aprovisionados e implementados todos los recursos, aparece el cuadro de diálogo Iniciar ejecución con una entrada de ejemplo similar a la siguiente.

```
{
    "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` indica al estado `Wait` que se repita cada 60 segundos.

- En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

  1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

     > **Note**
     >
     > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que

puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

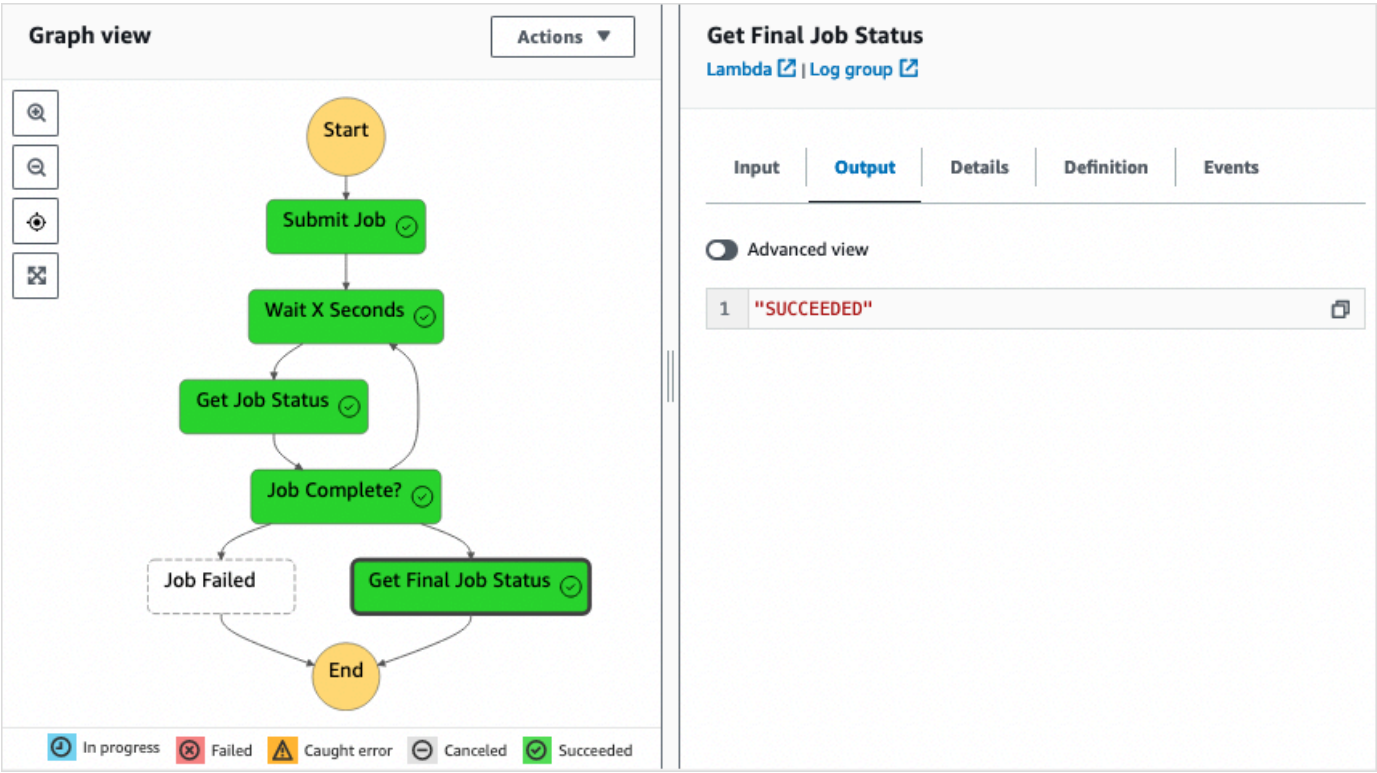
   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Por ejemplo, para ver el estado cambiante de su AWS Batch trabajo y los resultados en bucle de su ejecución, seleccione la pestaña Salida.

En la siguiente imagen se muestra el gráfico del estado de la ejecución en la Vista de gráfico. También muestra la salida de la ejecución del paso seleccionado en la pestaña Salida.



Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra AWS Lambda para enviar un AWS Batch trabajo. Examine este ejemplo de máquina de estados para ver cómo Step Functions controla Lambda y. AWS Batch

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
    "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
      }
```

```
    }
  }
```

Temporizador de tareas (Lambda, Amazon SNS)

Este proyecto de muestra crea un temporizador de tareas. Implementa una máquina de AWS Step Functions estados que implementa un `Wait` estado y usa una AWS Lambda función que envía una notificación del Amazon Simple Notification Service (Amazon SNS). Un estado Wait es un tipo de estado que espera a que un disparador realice una unidad de trabajo.

> **Note**
>
> Este proyecto de ejemplo implementa una AWS Lambda función para enviar una notificación del Amazon Simple Notification Service (Amazon SNS). También puede enviar una notificación de Amazon SNS directamente desde Amazon States Language. Consulte Uso AWS Step Functions con otros servicios.

Este proyecto de ejemplo crea la máquina de estados, una función Lambda y un tema de Amazon SNS, y configura los permisos AWS Identity and Access Management relacionados (IAM). Para obtener más información sobre los recursos que se crean con el proyecto de ejemplo Temporizador de tareas, consulte:

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

- AWS CloudFormation Guía del usuario
- Guía para desarrolladores de Amazon Simple Notification Service
- AWS Lambda Guía para desarrolladores
- Guía de introducción a IAM

Paso 1: Crear la máquina de estado y aprovisionar recursos

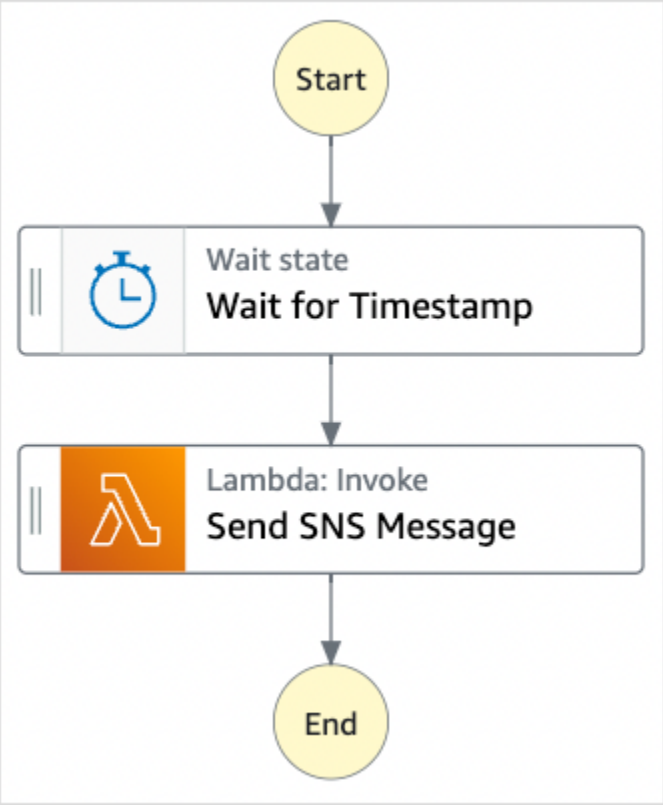
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Task Timer`** en el cuadro de búsqueda y, a continuación, seleccione Temporizador de tareas en los resultados de búsqueda que aparecen.

3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - una función de Lambda que envía una notificación de Amazon SNS.
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se muestra el gráfico del flujo de trabajo del proyecto de muestra Temporizador de tareas:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

  > **Important**
  >
  > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

Una vez aprovisionados e implementados todos los recursos, aparece el cuadro de diálogo Iniciar ejecución con una entrada de ejemplo similar a la siguiente.

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

  1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

     > **Note**
     >
     > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

  2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

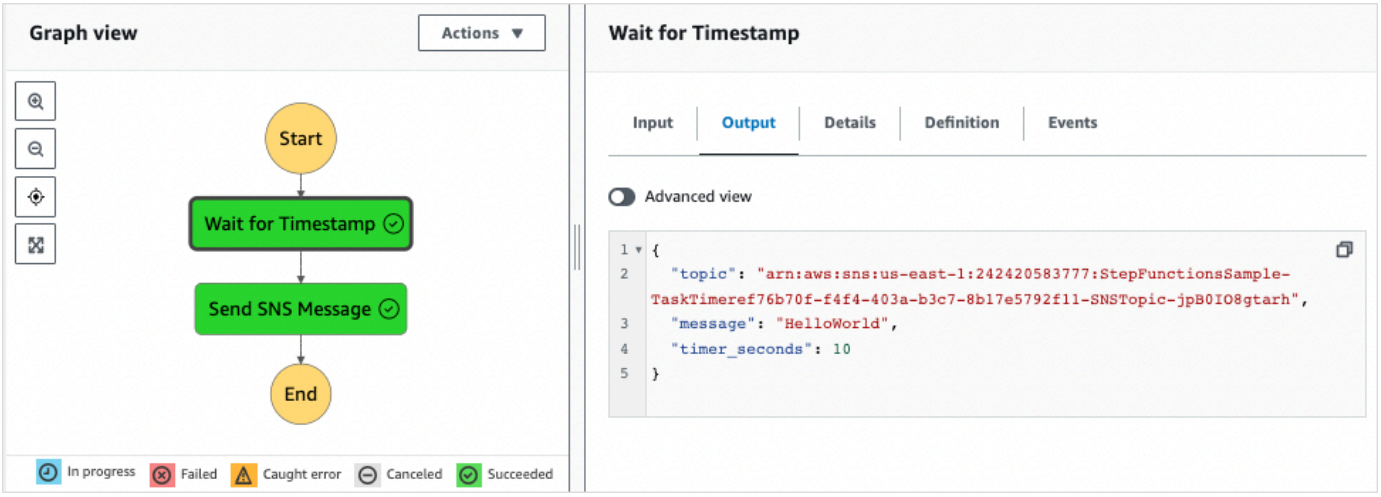
     Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

> **Note**
>
> Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Por ejemplo, en la siguiente imagen se muestra la salida del paso seleccionado Esperar a la marca temporal. La salida de este paso se transfiere como entrada al paso Enviar mensaje SNS.



Ejemplo de patrón de devolución de llamadas (Amazon SQS, Amazon SNS, Lambda)

Este proyecto de ejemplo muestra cómo hacer una AWS Step Functions pausa durante una tarea y esperar a que un proceso externo devuelva un token de tarea que se generó cuando se inició la tarea.

Cuando este proyecto de muestra se implementa y comienza una ejecución, se producen los siguientes pasos.

1. Step Functions pasa un mensaje que incluye un token de tarea a una cola de Amazon Simple Queue Service (Amazon SQS).
2. A continuación, Step Functions se detiene, esperando que ese token se devuelva.
3. La cola de Amazon SQS activa una AWS Lambda función que llama `SendTaskSuccess`con el mismo token de tarea.
4. Cuando se recibe el token de tarea, el flujo de trabajo continúa.
5. La tarea `"Notify Success"` publica un mensaje de Amazon Simple Notification Service (Amazon SNS) en el que se indica que se ha recibido la devolución de llamada.

Para obtener información acerca de cómo implementar el patrón de devolución de llamada en Step Functions, consulte Cómo esperar una devolución de llamada con el token de tarea.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

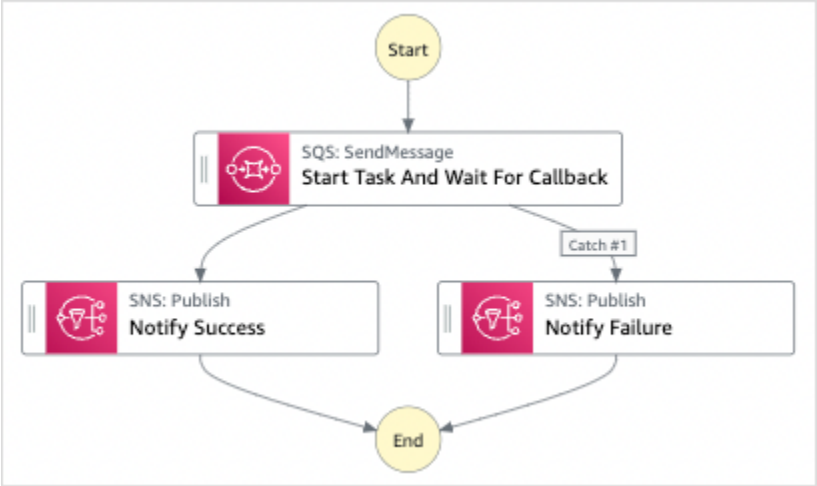
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Callback pattern example`** en el cuadro de búsqueda y, a continuación, seleccione Ejemplo de patrón de devolución de llamadas en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

En este proyecto de muestra se implementan los siguientes recursos:

- Una cola de mensajes de Amazon SQS.
- Función Lambda que llama a la acción de la API Step Functions. SendTaskSuccess
- Un tema de Amazon SNS para notificar el éxito o fracaso de una tarea e indicar si el flujo de trabajo puede continuar o no.
- Una máquina de AWS Step Functions estados
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Ejemplo de patrón de devolución de llamadas:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

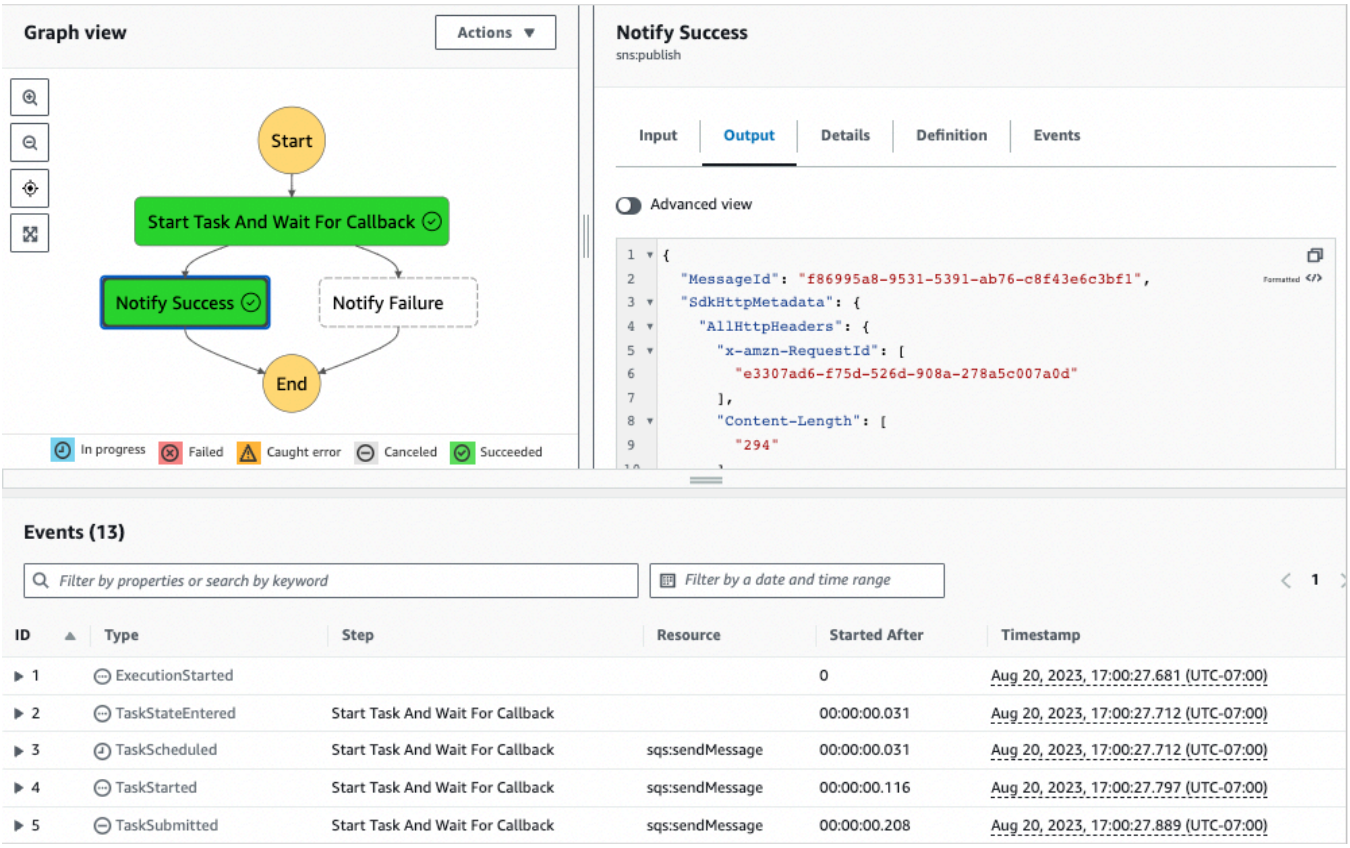
   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Por ejemplo, para revisar cómo ha progresado Step Functions a través del flujo de trabajo y recibido una llamada de devolución de Amazon SQS, revise las entradas de la tabla Eventos.

En la siguiente imagen se muestra el resultado de la ejecución del paso Notificar éxito. También muestra los cinco primeros eventos del historial de eventos de ejecución. Amplíe cada evento para ver más detalles sobre ese evento.



Ejemplo de devolución de llamada de Lambda

Para ver cómo funcionan juntos los componentes de este proyecto de ejemplo, consulta los recursos que se desplegaron en tu AWS cuenta. Por ejemplo, esta es la función de Lambda que llama a Step Functions con el token de tarea.

```
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);
        });
    }
};
```

Administrar un trabajo de Amazon EMR

Este proyecto de ejemplo demuestra Amazon EMR y AWS Step Functions su integración.

Enseña cómo crear un clúster de Amazon EMR, agregar varios pasos, ejecutarlos y después terminar el clúster.

> **Important**
>
> Amazon EMR no tiene una capa de precios gratuita. La ejecución del proyecto de muestra incurrirá en costos. Puede encontrar información acerca de los precios en la página Precios de Amazon EMR. La disponibilidad de la integración de servicios de Amazon EMR depende de la disponibilidad de las API de Amazon EMR. Por este motivo, es posible que este proyecto de ejemplo no funcione correctamente en algunas AWS regiones. Consulte la documentación de Amazon EMR para conocer las limitaciones en regiones especiales.

Paso 1: Crear la máquina de estado y aprovisionar recursos

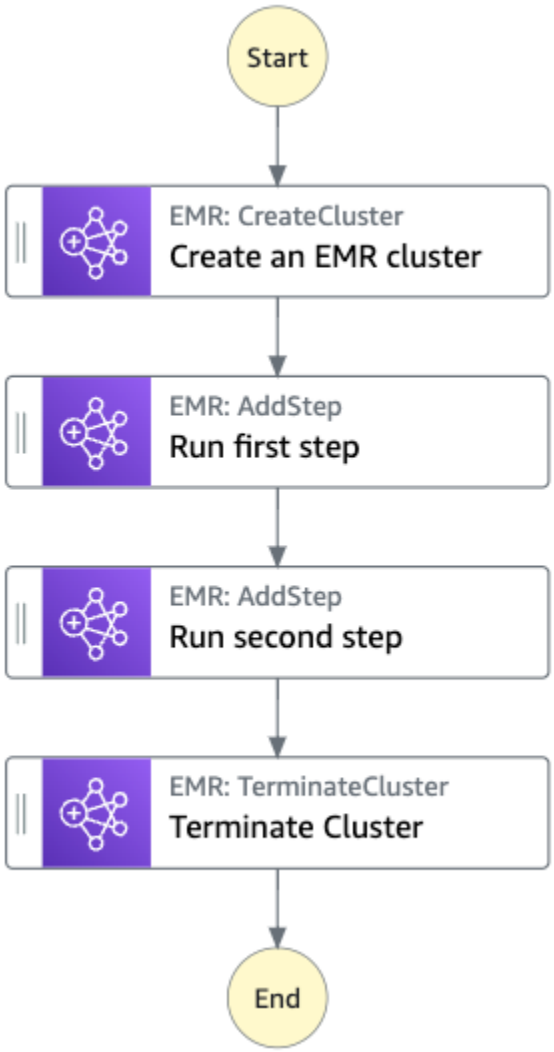
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

2. Escriba **`Manage an EMR job`** en el cuadro de búsqueda y, a continuación, seleccione Administración de un trabajo de EMR en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Un bucket de Amazon S3
   - Un clúster Amazon EMR
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Administración de un trabajo de EMR:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con Amazon EMR pasando parámetros directamente a esos recursos. Examine esta máquina de estado de ejemplo para ver cómo Step Functions utiliza una máquina de estado para llamar a la tarea de Amazon EMR de forma síncrona, espera a que la tarea se realice correctamente o falle y entonces termina el clúster.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Es una práctica recomendada incluir solo los permisos necesarios en las políticas de IAM.

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

La siguiente política garantiza que `addStep` tenga los permisos suficientes.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejecutar un trabajo de EMR Serverless

En este proyecto de muestra se ilustra cómo se puede crear e iniciar una aplicación de EMR Serverless. Este proyecto también muestra cómo se pueden ejecutar varios trabajos dentro de esa aplicación.

Este proyecto de ejemplo crea la máquina de estados, los AWS recursos auxiliares y configura los permisos de IAM relacionados. Explore este proyecto de muestra para aprender acerca de la ejecución de trabajos de EMR Serverless mediante máquinas de estado de Step Functions o úselo como punto de partida para sus propios proyectos.

> **Important**
>
> EMR Serverless no tiene una capa de precios gratuita. La ejecución del proyecto de muestra incurrirá en costos. Puede encontrar información acerca de los precios en la página Precios de Amazon EMR Serverless.
> Además, la disponibilidad de la integración de servicios de EMR Serverless está sujeta a la disponibilidad de las API de EMR Serverless. Por ello, es posible que este proyecto de

ejemplo no funcione correctamente en algunas Regiones de AWS. Consulte el tema Otras consideraciones para obtener información sobre la disponibilidad de EMR Serverless en Regiones de AWS.

Plantilla de AWS CloudFormation y recursos adicionales

Utilice una plantilla de CloudFormation para implementar este proyecto de muestra. Esta plantilla crea los siguientes recursos en su: Cuenta de AWS

- Una máquina de estado de Step Functions.
- Rol de ejecución para la máquina de estado. Esta función concede los permisos que su máquina de estado necesita para acceder a otros Servicios de AWS recursos, como la EMR Serverless CreateApplicationacción.
- Rol de ejecución de de trabajos para EMR Serverless. Este rol concede los permisos que puede asumir una ejecución de trabajo de EMR Serverless cuando llame a otros servicios en su nombre.

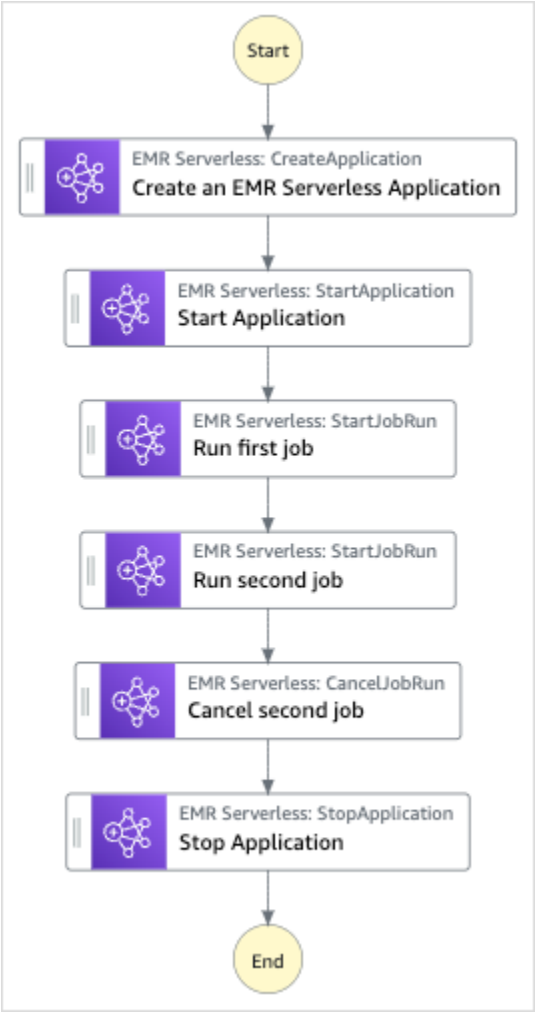
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`EMR Serverless`** en el cuadro de búsqueda y, a continuación, seleccione Ejecutar un trabajo de EMR Serverless en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una máquina de estado de Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

   En la siguiente imagen se ilustra el gráfico de flujo del trabajo del proyecto de muestra Ejecutar un trabajo de EMR Serverless:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Iniciar un flujo de trabajo dentro de un flujo de trabajo (Step Functions, Lambda)

Este proyecto de ejemplo muestra cómo utilizar una máquina de AWS Step Functions estados para iniciar otras ejecuciones de máquinas de estados. Para obtener información sobre cómo iniciar las ejecuciones de una máquina de estado desde otra máquina de estado, consulte Iniciar ejecuciones de flujo de trabajo desde un estado de tarea.

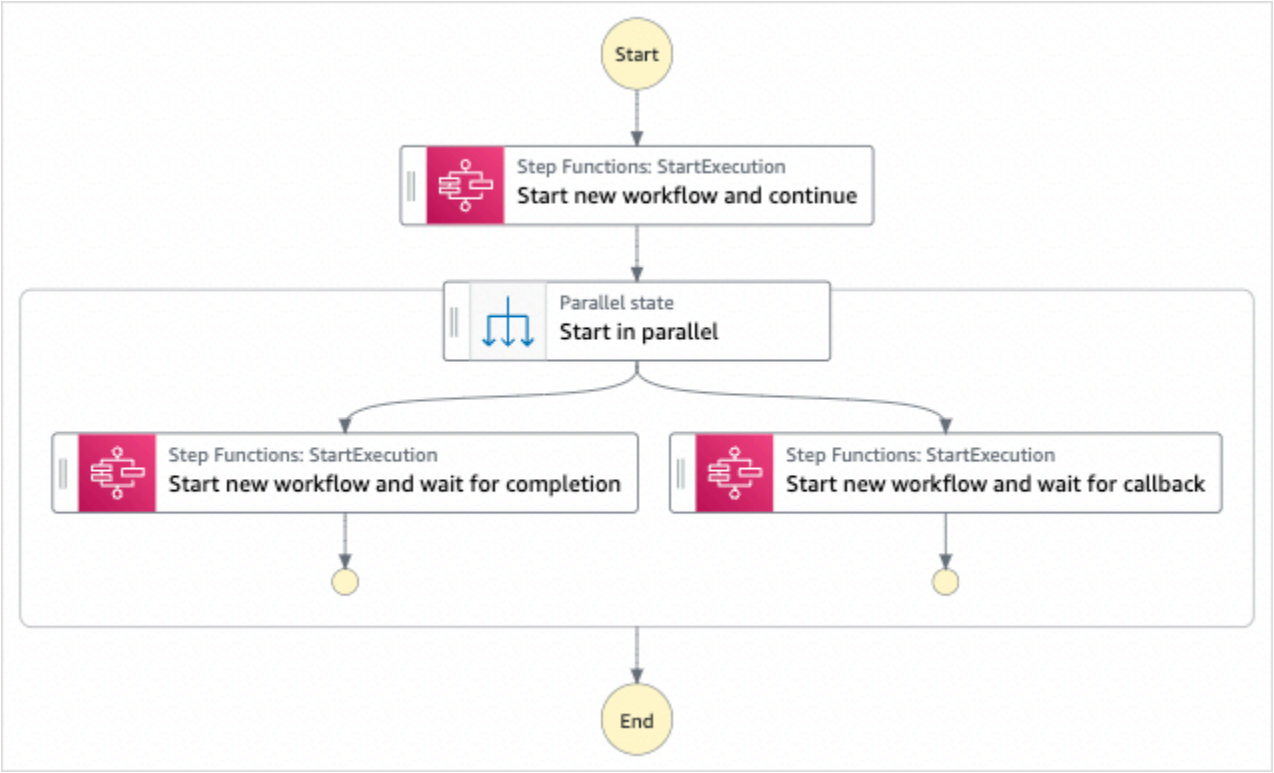
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Start a workflow within a workflow`** en el cuadro de búsqueda y, a continuación, seleccione Iniciar un flujo de trabajo dentro de un flujo de trabajo en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una máquina de estado adicional La ejecución de esta máquina de estado la inicia la máquina de estado que usted ejecuta.
   - Una función de Lambda de devolución de llamada. Esta función se utiliza en la máquina de estado adicional para implementar el mecanismo de devolución de llamada.
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se muestra el gráfico de flujo de trabajo del proyecto de muestra Iniciar un flujo de trabajo dentro de un flujo de trabajo:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo integra otra máquina de estados pasando AWS Lambda los parámetros directamente a esos recursos.

Examine esta máquina de estado de ejemplo para ver cómo Step Functions llama a la acción de la API `StartExecution` para la otra máquina de estado. Lanza dos instancias de la otra máquina de estado en paralelo: una con el patrón Ejecutar un trabajo (.sync) y otra con el patrón Cómo esperar una devolución de llamada con el token de tarea.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Procesamiento dinámico de datos con un estado Map

Este proyecto de muestra demuestra el paralelismo dinámico mediante un estado Map.

En este proyecto, Step Functions utiliza una AWS Lambda función para extraer mensajes de una cola de Amazon SQS y pasar una matriz JSON de esos mensajes a un estado. `Map` Para cada mensaje de la cola, la máquina de estado escribe el mensaje en DynamoDB, invoca la otra función de Lambda para eliminar el mensaje de Amazon SQS y, a continuación, publica el mensaje en el tema de Amazon SNS.

Para obtener más información acerca de los estados `Map` y las integraciones de los servicios de Step Functions, consulte los temas siguientes:

- Map
- Uso AWS Step Functions con otros servicios

Paso 1: Crear la máquina de estado y aprovisionar recursos

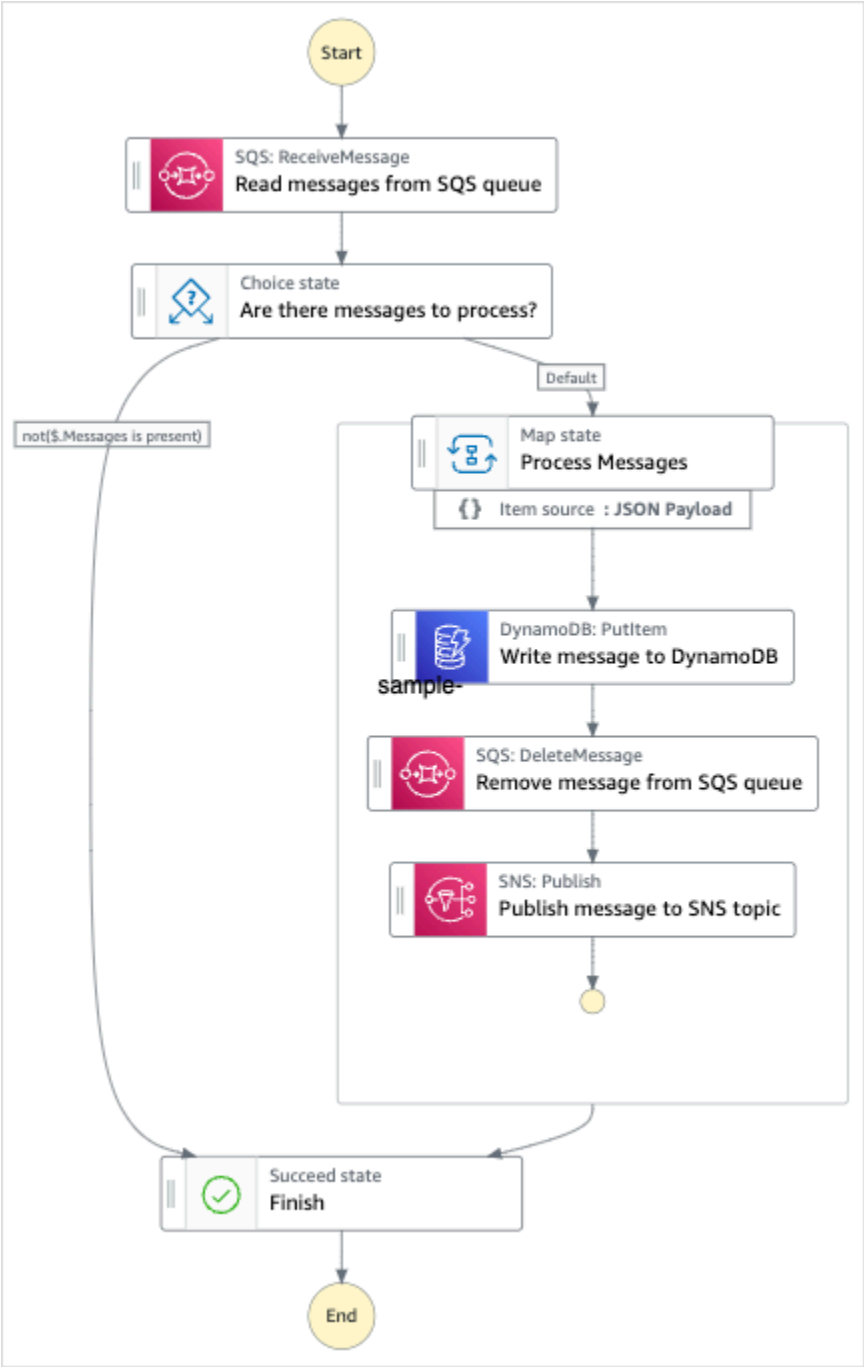
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Dynamically process data with a Map state`** en el cuadro de búsqueda y, a continuación, seleccione Procesamiento dinámico de datos con un estado Map en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una cola de Amazon SQS en la que el estado Map lee y elimina los mensajes de forma iterativa.
   - Una tabla de DynamoDB en la que el estado Map escribe mensajes de forma iterativa.
   - Un tema de Amazon SNS en el que Step Functions publica los mensajes que lee de la cola de Amazon SQS.
   - Dos AWS Lambda funciones

- Una máquina AWS Step Functions de estados
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Procesamiento dinámico de datos con un estado Map:



5. Elija Utilizar plantilla para continuar con la selección.

6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

     > **Tip**
     >
     > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

     Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

     El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

     Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Se pueden aplicar cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Después de implementar los recursos del proyecto de muestra, tendrá que añadir elementos a la cola de Amazon SQS y suscribirse al tema de Amazon SNS para poder iniciar una ejecución de la máquina de estado.

Paso 2: Suscribirse al tema de Amazon SNS

1. Abra la consola de Amazon SNS.
2. Seleccione Temas y elija el tema creado por el proyecto de muestra del estado `Map`.

   El nombre será similar a MapSampleProj-SNSTopic-1cQo4HQ3IR1kN.
3. Seleccione Crear suscripción.

   Se muestra la página Crear suscripción, que muestra el valor de ARN del tema del tema.
4. En Protocolo, elija Correo electrónico.
5. En Punto de conexión, escriba una dirección de correo electrónico para suscribirse al tema.
6. Seleccione Crear suscripción.

   > **Note**
   >
   > Debe confirmar la suscripción en su correo electrónico para que esta se active.
7. Abra el correo electrónico Confirmación de suscripción en la cuenta relacionada y abra la URL Confirmar suscripción.

   Se muestra la página Suscripción confirmada.

Paso 3: Añadir mensajes a la cola de Amazon SQS

1. Abra la consola de Amazon SQS.
2. Seleccione la cola creada por el proyecto de muestra del estado `Map`.

El nombre será similar a MapSampleProj-SqSqueue-1uDiC9vZdOrn7.

3. Seleccione Enviar y recibir mensajes.
4. En la página Enviar un mensaje, introduzca un mensaje y seleccione Enviar mensaje.
5. Introduzca otro mensaje y seleccione Enviar mensaje. Siga escribiendo más mensajes hasta que tenga varios en la cola de Amazon SQS.

Paso 4: Ejecutar la máquina de estado

> **Note**
>
> Las colas de Amazon SNS son de consistencia final. Para obtener los mejores resultados, espere unos minutos entre llenar la cola y llevar a cabo una ejecución de la máquina de estado.

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

> **Note**
>
> Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con Amazon SQS, Amazon SNS y Lambda pasando parámetros directamente a esos recursos.

Examine este ejemplo de máquina de estado para ver cómo Step Functions controla Lambda, DynamoDB y Amazon SNS conectándose al nombre de recurso de Amazon (ARN) del campo `Resource` y pasando `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

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

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Procesar un archivo CSV con Distributed Map

En este proyecto de muestra se ilustra cómo se puede utilizar el estado Distributed Map para iterar más de 10 000 filas de un archivo CSV que se genera mediante una función de Lambda. El archivo CSV contiene información de envío de los pedidos de los clientes y se almacena en un bucket de Amazon S3. Distributed Map itera sobre un lote de 10 filas en el archivo CSV para el análisis de los datos.

Distributed Map contiene una función de Lambda para detectar cualquier pedido retrasado. Distributed Map también contiene un Inline Map para procesar los pedidos retrasados en un lote y devolver estos pedidos retrasados en una matriz. Para cada pedido retrasado, Inline Map envía un mensaje a una cola de Amazon SQS. Por último, este proyecto de muestra almacena los resultados de Map Run en otro bucket de Amazon S3 de su Cuenta de AWS.

Con Distributed Map, puede ejecutar hasta 10 000 ejecuciones paralelas de flujos de trabajo secundarios a la vez. En este proyecto de muestra, la simultaneidad máxima de Distributed Map se establece en 1000, lo que la limita a 1000 ejecuciones paralelas de flujos de trabajo secundarios.

Este proyecto de ejemplo crea la máquina de estados, los AWS recursos auxiliares y configura los permisos de IAM relacionados. Explore este proyecto de muestra para aprender acerca del uso de Distributed Map para orquestar cargas de trabajo paralelas a gran escala o úselo como punto de partida para sus propios proyectos.

AWS CloudFormation plantilla y recursos adicionales

Utiliza una CloudFormation plantilla para implementar este proyecto de ejemplo. Esta plantilla crea los siguientes recursos en su Cuenta de AWS:

- Una máquina de estado de Step Functions.
- Rol de ejecución para la máquina de estado. Esta función otorga los permisos que su máquina de estado necesita para acceder a otros Servicios de AWS recursos, como la acción Invoke de la función Lambda.
- Una función de Lambda denominada `CSVGeneratorFunction` que genera un archivo CSV que contiene los detalles del pedido del cliente.
- Un rol de ejecución para la función de Lambda del generador CSV. Este rol otorga a la función permiso para acceder a otros. Servicios de AWS
- Un bucket de entrada de Amazon S3 para almacenar el archivo CSV generado.

- Una función de Lambda de detección de pedidos retrasados que analiza los datos del archivo CSV y detecta cualquier pedido retrasado.
- Un rol de ejecución para la función de Lambda de pedidos retrasados. Este rol otorga a la función permiso para acceder a otros Servicios de AWS.
- Un bucket de salida de Amazon S3 para almacenar los resultados del análisis de los pedidos de los clientes.
- Una cola de Amazon SQS a la que Step Functions envía mensajes por cada pedido retrasado. Estos mensajes contienen los ID de los clientes y sus pedidos.
- Un grupo de CloudWatch registros que almacena información relacionada con el historial de ejecución de la máquina de estados.

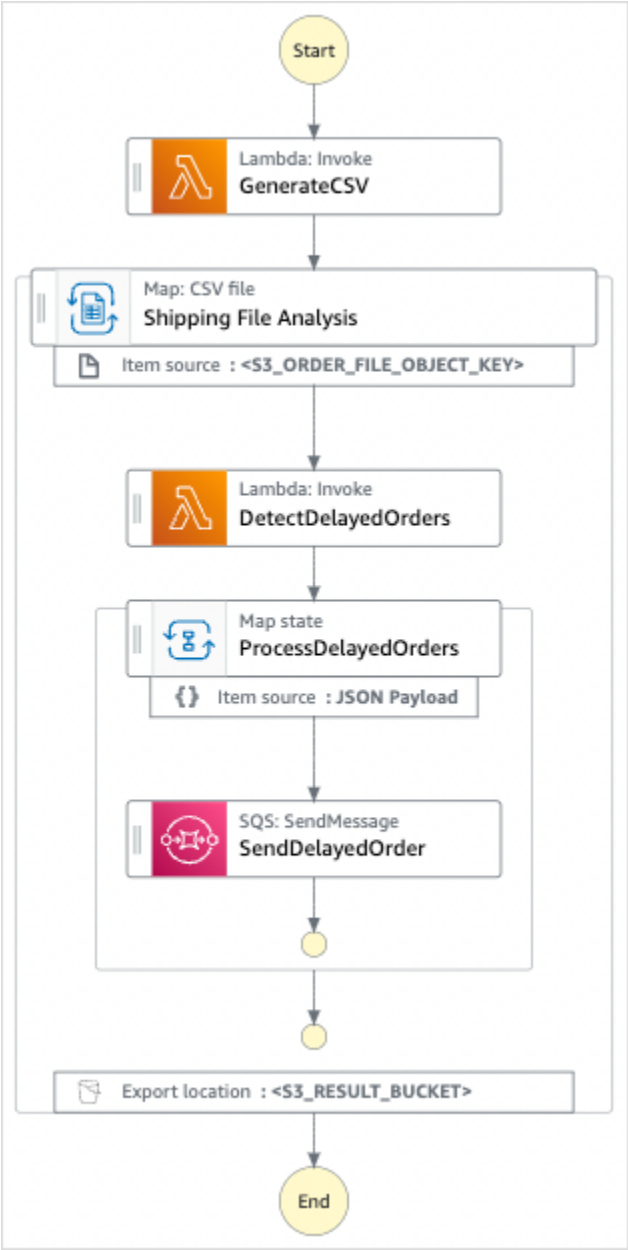
> **Important**
>
> Se aplican cargos estándar por cada servicio.

Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Distributed Map to process a CSV file in S3`** en el cuadro de búsqueda y, a continuación, seleccione Asignación distribuida para procesar un archivo CSV en S3 en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   Para obtener información sobre los recursos que se crearán para este proyecto de muestra, consulte AWS CloudFormation plantilla y recursos adicionales.

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Asignación distribuida para procesar un archivo CSV en S3:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar

la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

Una vez aprovisionados e implementados todos los recursos, puede ejecutar la máquina de estado.

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Introduzca los valores de entrada en formato JSON para ejecutar el proyecto de muestra.

      Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

      > **Note**
      >
      > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

   b. Seleccione Iniciar ejecución.

   c. (Opcional) La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Una vez que se complete la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente.

      - Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.
      - Para obtener más información sobre cómo ver la ejecución de un estado Distributed Map en la consola, consulte Examen de Map Run.

   d. (Opcional) Revise los resultados de ejecución exportados al bucket de Amazon S3. Estos resultados incluyen datos, como la entrada y salida de la ejecución, el ARN y el estado de la ejecución. Para obtener más información, consulte ResultWriter.

Procesar datos de un bucket de Amazon S3 con Distributed Map

En este proyecto de muestra se explica cómo puede utilizar el estado Distributed Map para procesar datos a gran escala, por ejemplo, analizar datos meteorológicos históricos e identificar la estación meteorológica que tiene la temperatura media más elevada del planeta cada mes. Los datos meteorológicos se registran en más de 12 000 archivos CSV, que a su vez se almacenan en un bucket de Amazon S3.

Este proyecto de muestra incluye dos estados Distributed Map denominados Distributed S3 copy NOA Data y ProcessNOAAdata. Los datos NOA de copia distribuida de S3 recorren los archivos CSV de un bucket público de Amazon S3 denominado noaa-gsod-pdsy los copian en un bucket de Amazon S3 del suyo. Cuenta de AWS ProcessNOAAData itera sobre los archivos copiados e incluye una función de Lambda que realiza el análisis de la temperatura.

El proyecto de ejemplo comprueba primero el contenido del bucket de Amazon S3 con una llamada a la acción de la API ListObjectsV2. Según el número de claves devueltas en respuesta a esta llamada, el proyecto de muestra toma una de las siguientes decisiones:

- Si el recuento de claves es mayor o igual a 1, el proyecto pasa al estado ProcessNOAAData. Este estado del mapa distribuido incluye una Lambda función denominada TemperatureFunctionque busca la estación meteorológica que tuvo la temperatura media más alta cada mes. Esta función devuelve un diccionario con `year-month` como clave y un diccionario que contiene información sobre la estación meteorológica como valor.
- Si el recuento de claves devuelto no supera 1, el estado de datos NOA de la copia distribuida de S3 muestra todos los objetos del depósito público noaa-gsod-pdsy copia de forma iterativa los objetos individuales en otro depósito de su cuenta en lotes de 100. Una Inline Map realiza la copia iterativa de los objetos.

  Una vez copiados todos los objetos, el proyecto pasa al estado ProcessNOAAData para procesar los datos meteorológicos.

El proyecto de muestra finalmente pasa a una Lambda función reductora que realiza una agregación final de los resultados devueltos por la TemperatureFunctionfunción y los escribe en una tabla. Amazon DynamoDB

Con Distributed Map, puede ejecutar hasta 10 000 ejecuciones paralelas de flujos de trabajo secundarios a la vez. En este proyecto de muestra, la simultaneidad máxima de ProcessNOAAData

de Distributed Map se establece en 3000, lo que la limita a 3000 ejecuciones de flujos de trabajo secundarios paralelos.

Este proyecto de ejemplo crea la máquina de estados, los AWS recursos auxiliares y configura los permisos de IAM relacionados. Explore este proyecto de muestra para aprender acerca del uso de Distributed Map para orquestar cargas de trabajo paralelas a gran escala o úselo como punto de partida para sus propios proyectos.

> **Important**
>
> Este proyecto de muestra solo está disponible en la región Este de EE. UU. (Norte de Virginia)

AWS CloudFormation plantilla y recursos adicionales

Utiliza una CloudFormation plantilla para implementar este proyecto de ejemplo. Esta plantilla crea los siguientes recursos en su Cuenta de AWS:

- Una máquina de estado de Step Functions.
- Rol de ejecución para la máquina de estado. Esta función otorga los permisos que su máquina de estado necesita para acceder a otros Servicios de AWS recursos, como la acción Invoke de la función Lambda.
- Un bucket de Amazon S3 denominado `NOAADataBucket`. Este bucket contiene los archivos CSV con datos meteorológicos.
- Una función de Lambda denominada `ReducerFunction` que realiza una agregación final de los datos meteorológicos y escribe los resultados en una tabla de Amazon DynamoDB.
- Rol de ejecución para la función de Lambda reductora. Este rol otorga a la función permiso para acceder a otros. Servicios de AWS
- Un bucket de salida de Amazon S3 denominado `ResultsBucket` para almacenar los resultados del análisis meteorológico.
- Una tabla de DynamoDB denominada `ResultsDynamoDBTable` que contiene los resultados devueltos por `ReducerFunction`.
- Una función de Lambda denominada `TemperatureFunction` que encuentra la temperatura media mensual más elevada.

- Rol de ejecución para la función de Lambda. Este rol otorga a la función permiso para acceder a otros Servicios de AWS.
- Un grupo de CloudWatch registros que almacena información relacionada con el historial de ejecución de la máquina de estados.

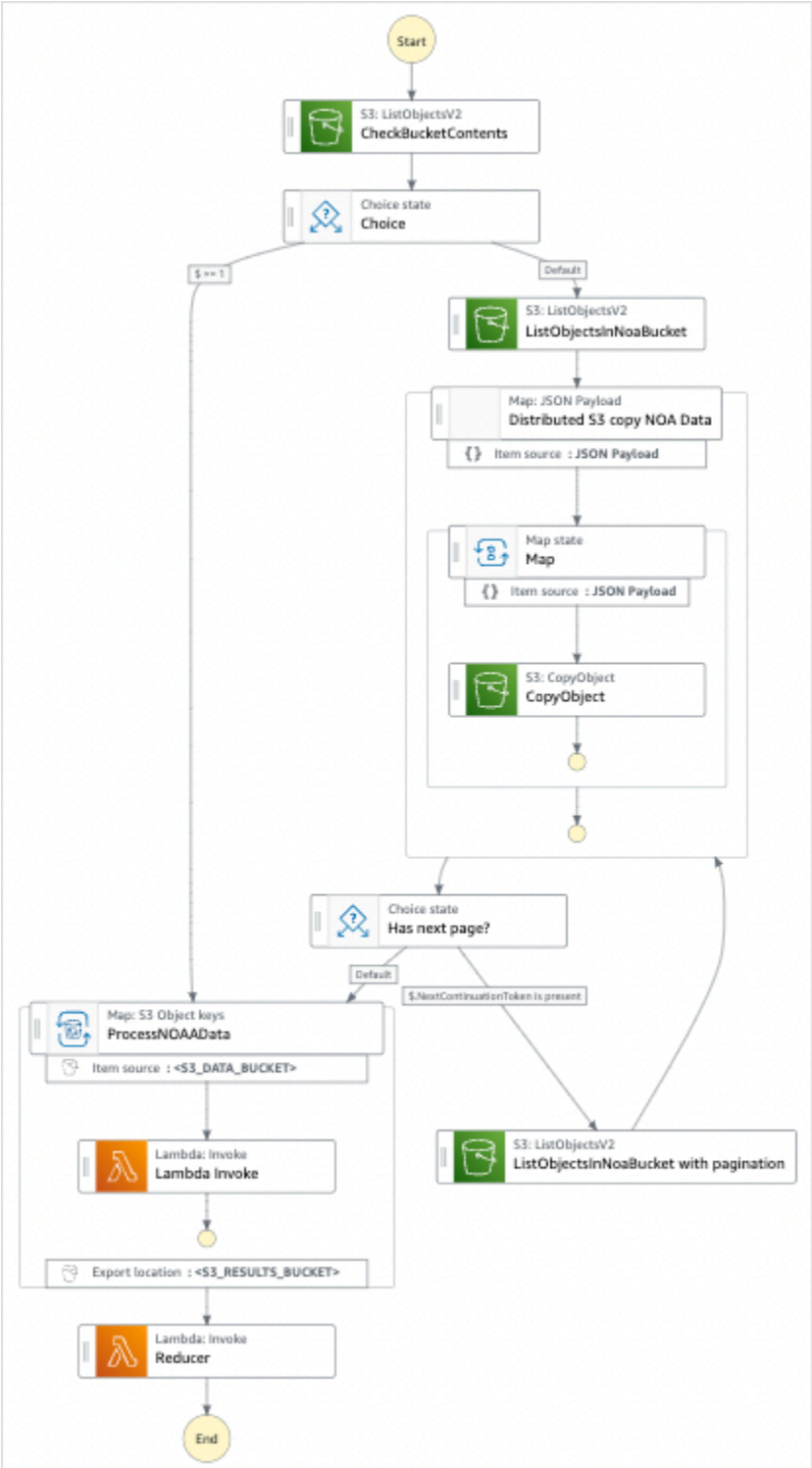
> **Important**
>
> Se aplican cargos estándar por cada servicio.

Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Distributed Map to process files in S3`** en el cuadro de búsqueda y, a continuación, seleccione Asignación distribuida para procesar archivos en S3 en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   Para obtener información sobre los recursos que se crearán para este proyecto de muestra, consulte AWS CloudFormation plantilla y recursos adicionales.

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Asignación distribuida para procesar archivos en S3:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

  > **Important**
  >
  > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

Una vez aprovisionados e implementados todos los recursos, puede ejecutar la máquina de estado.

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   a. (Opcional) Introduzca los valores de entrada en formato JSON para ejecutar el proyecto de muestra.

      Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

      > **Note**
      >
      > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

   b. Seleccione Iniciar ejecución.

   c. (Opcional) La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Una vez que se complete la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente.

- Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.
- Para obtener más información sobre cómo ver la ejecución de un estado Distributed Map en la consola, consulte Examen de Map Run.

d. (Opcional) Revise los resultados de ejecución exportados al bucket de Amazon S3. Estos resultados incluyen datos, como la entrada y salida de la ejecución, el ARN y el estado de la ejecución. Para obtener más información, consulte ResultWriter.

Entrenamiento de un modelo de aprendizaje automático

Este proyecto de ejemplo demuestra cómo usar SageMaker y AWS Step Functions entrenar un modelo de aprendizaje automático y cómo transformar por lotes un conjunto de datos de prueba.

En este proyecto, Step Functions utiliza una función de Lambda para propagar datos en un bucket de Amazon S3 con un conjunto de datos de prueba. A continuación, entrena un modelo de aprendizaje automático y realiza una transformación por lotes mediante la integración SageMaker de servicios.

Para obtener más información sobre SageMaker las integraciones de los servicios Step Functions, consulte lo siguiente:

- Uso AWS Step Functions con otros servicios
- Administre SageMaker con Step Functions

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y la capa gratuita, consulta SageMaker los precios.

Paso 1: Crear la máquina de estado y aprovisionar recursos

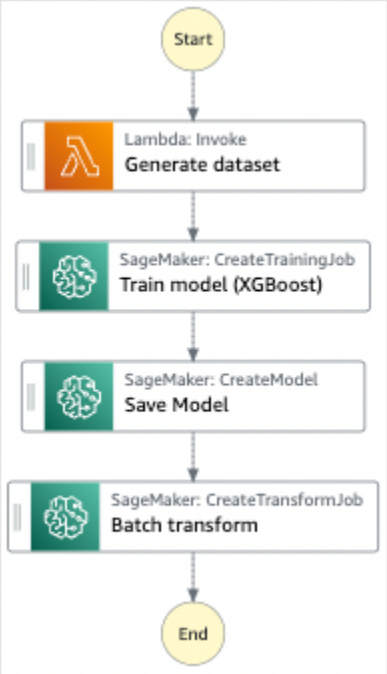
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.

2. Escriba **`Train a machine learning model`** en el cuadro de búsqueda y, a continuación, seleccione Entrenamiento de un modelo de machine learning en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - ¿Una AWS Lambda función
   - Un bucket de Amazon Simple Storage Service (Amazon S3)
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico de flujo del trabajo del proyecto de muestra Formación de un modelo de machine learning:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

  > **Important**
  >
  > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con esos recursos SageMaker y AWS Lambda los transfiere directamente a ellos, y utiliza un bucket de Amazon S3 para la fuente y la salida de los datos de entrenamiento.

Examine este ejemplo de máquina de estados para ver cómo Step Functions controla Lambda y. SageMaker

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los

recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

La siguiente política permite a la función de Lambda propagar los datos de muestra en el bucket de Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Ajuste de un modelo de aprendizaje automático

Este proyecto de ejemplo muestra cómo SageMaker ajustar los hiperparámetros de un modelo de aprendizaje automático y transformar por lotes un conjunto de datos de prueba.

En este proyecto, Step Functions utiliza una función de Lambda para propagar datos en un bucket de Amazon S3 con un conjunto de datos de prueba. A continuación, crea un trabajo de ajuste de hiperparámetros mediante la integración de SageMakerservicios. A continuación, utiliza una función Lambda para extraer la ruta de datos, guarda el modelo de ajuste, extrae el nombre del modelo y, a continuación, ejecuta un trabajo de transformación por lotes para realizar inferencias. SageMaker

Para obtener más información sobre SageMaker las integraciones de los servicios Step Functions, consulte lo siguiente:

- Uso AWS Step Functions con otros servicios
- Administre SageMaker con Step Functions

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y la capa gratuita, consulta SageMakerlos precios.

Paso 1: Crear la máquina de estado y aprovisionar recursos

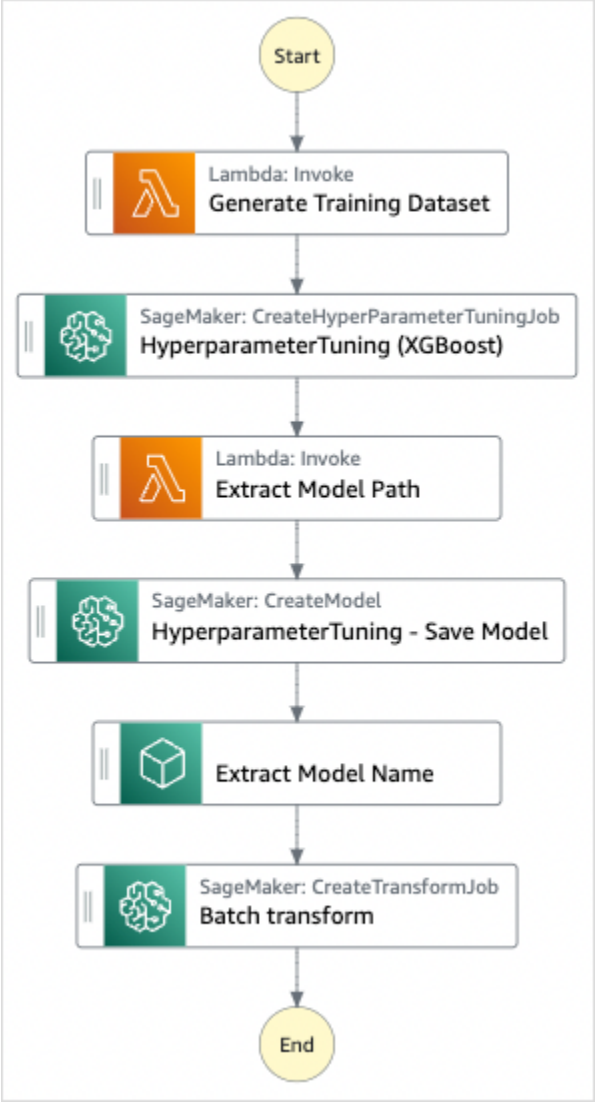
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Tune a machine learning model`** en el cuadro de búsqueda y, a continuación, seleccione Ajuste de un modelo de aprendizaje automático en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de

muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

En este proyecto de muestra se implementan los siguientes recursos:

- Tres AWS Lambda funciones
- Un bucket de Amazon Simple Storage Service (Amazon S3)
- Una máquina AWS Step Functions de estados
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Ajuste de un modelo de aprendizaje automático:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

     > **Tip**
     >
     > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

     Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

     El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

> **Note**
>
> Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con esos recursos SageMaker y AWS Lambda los transfiere directamente a ellos, y utiliza un bucket de Amazon S3 para la fuente y la salida de los datos de entrenamiento.

Examine este ejemplo de máquina de estados para ver cómo Step Functions controla Lambda y. SageMaker

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
    "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
        }
    }
}
```

Para obtener información acerca de cómo configurar IAM cuando se utiliza Step Functions con otros servicios de AWS , consulte Políticas de IAM para servicios integrados.

Ejemplos de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

La siguiente política de IAM se adjunta a la máquina de estado y permite que la ejecución de la máquina de estado acceda a los SageMaker recursos necesarios de Lambda y Amazon S3.

```
{
    "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"
        }
    ]
```

```
}
```

Se hace referencia a la siguiente política de IAM en los campos `TrainingJobDefinition` y `HyperparameterTuning` del estado `HyperparameterTuning`.

```
{
    "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"
        }
```

```
    ]
}
```

La siguiente política de IAM permite a la función de Lambda propagar los datos de muestra en el bucket de Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

Para obtener información acerca de cómo configurar IAM cuando se utiliza Step Functions con otros servicios de AWS , consulte Políticas de IAM para servicios integrados.

Procesar mensajes de un volumen elevado desde Amazon SQS (flujos de trabajo rápidos)

Este proyecto de ejemplo demuestra cómo utilizar un flujo de AWS Step Functions trabajo rápido para procesar mensajes o datos de una fuente de eventos de gran volumen, como Amazon Simple Queue Service (Amazon SQS). Dado que los flujos de trabajo rápidos se pueden iniciar a una velocidad muy elevada, son ideales para las cargas de trabajo de datos de streaming o procesamiento de eventos de un volumen elevado.

A continuación se muestran dos métodos utilizados con frecuencia para ejecutar la máquina de estado desde un origen de eventos:

- Configure una regla de Amazon CloudWatch Events para iniciar la ejecución de una máquina de estados siempre que la fuente del evento emita un evento. Para obtener más información, consulte Creación de una regla de CloudWatch eventos que se active en un evento.

- Asigne el origen de eventos a una función Lambda y escriba el código de función para ejecutar la máquina de estado. La AWS Lambda función se invoca cada vez que la fuente de eventos emite un evento, lo que a su vez inicia una ejecución en una máquina de estados. Para obtener más información, consulte Uso de AWS Lambda con Amazon SQS.

Este proyecto de ejemplo utiliza el segundo método para iniciar una ejecución cada vez que la cola de Amazon SQS envía un mensaje. Puede usar una configuración similar para activar la ejecución de flujos de trabajo rápidos desde otros orígenes de eventos, como Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB y Amazon Kinesis.

Para obtener más información acerca de los flujos de trabajo rápidos y las integraciones de servicios de Step Functions, consulte los siguientes temas:

- Flujos de trabajo estándar en comparación con flujos de trabajo rápidos
- Uso AWS Step Functions con otros servicios
- Cuotas

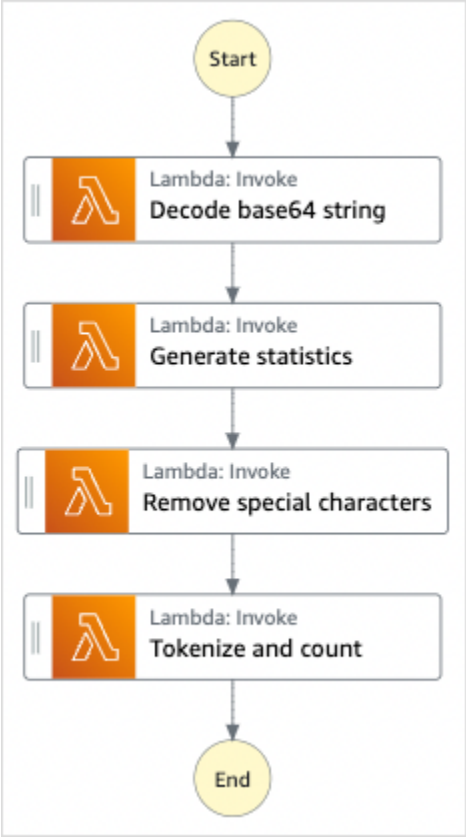
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Process high-volume messages from SQS`** en el cuadro de búsqueda y, a continuación, seleccione Procesar mensajes de un volumen elevado desde SQS en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Cuatro funciones de Lambda
   - Una cola de Amazon SQS
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el proyecto de muestra Procesar mensajes de un volumen elevado desde SQS:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Activar la ejecución de la máquina de estado

1. Abra la consola de Amazon SQS.
2. Seleccione la cola que ha creado el proyecto de ejemplo.

El nombre será similar a Example-SQSQueue-wJalrXUtnFEMI.

3. En la lista Acciones de cola, seleccione Enviar un mensaje.
4. Utilice el botón de copiar para copiar el siguiente mensaje y, en la ventana Enviar un mensaje, escríbalo y seleccione el botón Enviar mensaje.

> **Note**
>
> En este mensaje de ejemplo, la línea de `input:` tiene un formato con saltos de línea para ajustarse a la página. Utilice el botón de copiar o asegúrese de que se introduzca como una línea única sin saltos de línea.

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc

  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw

  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT

  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU

  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic

  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB

  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBzdWF0aW5nIG1pb
}
```

5. Elija Close.
6. Abra la consola de Step Functions.
7. Ve a tu grupo de CloudWatch registros de Amazon Logs e inspecciona los registros. El nombre del grupo de registros tendrá el siguiente aspecto: ExpressLogGroup -wjalRxUtnFemi.

Código de función de Lambda de ejemplo

El siguiente es el código de la función Lambda que muestra cómo la función Lambda iniciadora inicia la ejecución de una máquina de estados mediante el SDK. AWS

```
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
    )
```

Código de la máquina de estado de ejemplo

El flujo de trabajo rápido de este proyecto de ejemplo está compuesto por un conjunto de funciones Lambda para procesar textos.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
  "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
    }
  }
}
```

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

La siguiente política garantiza que haya permisos suficientes para CloudWatch los registros.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Ejemplo de punto de comprobación selectivo (flujos de trabajo rápidos)

Este proyecto de ejemplo muestra cómo combinar flujos de trabajo estándar y rápidos mediante la ejecución de un flujo de trabajo de comercio electrónico simulado que realiza puntos de comprobación selectivos. Al implementar este proyecto de ejemplo se creará una máquina de estado de flujos de trabajo estándar, una máquina de estado de flujos de trabajo rápidos anidados, una función de AWS Lambda , una cola de Amazon Simple Queue Service (Amazon SQS) y un tema de Amazon Simple Notification Service (Amazon SNS).

Para obtener más información acerca de los flujos de trabajo rápidos, flujos de trabajo anidados y las integraciones de servicios de Step Functions, consulte los siguientes temas.

- Flujos de trabajo estándar en comparación con flujos de trabajo rápidos
- Iniciar ejecuciones de flujo de trabajo desde un estado de tarea
- Uso AWS Step Functions con otros servicios

Paso 1: Crear la máquina de estado y aprovisionar recursos

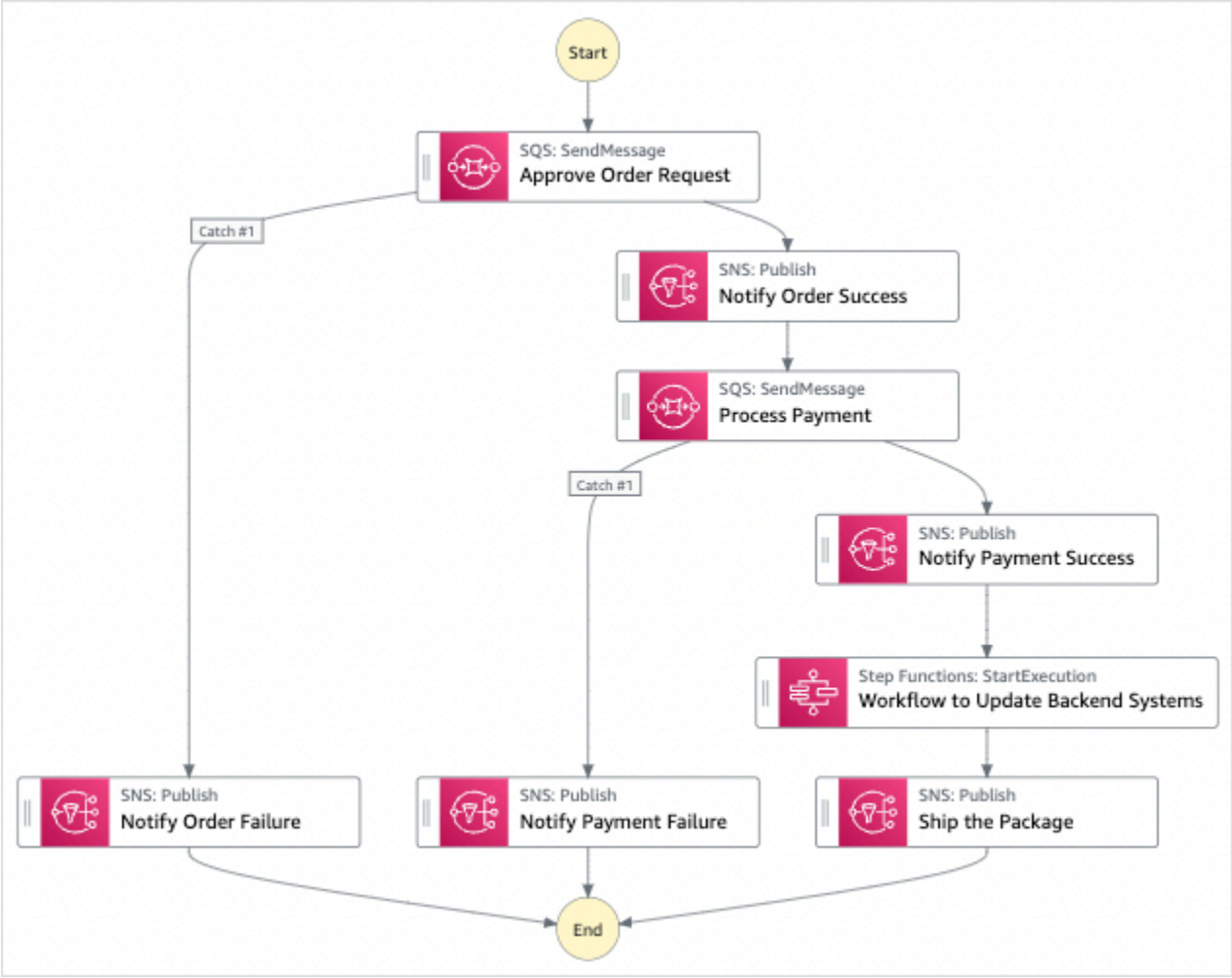
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Selective checkpointing example`** en el cuadro de búsqueda y, a continuación, seleccione Ejemplo de punto de comprobación selectivo en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - ¿Una AWS Lambda función
   - Una cola de Amazon SQS
   - Un tema de Amazon SNS

- Una máquina de AWS Step Functions estados del tipo Standard
- Una máquina de estado de Step Functions de tipo rápido
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Punto de comprobación selectivo:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

> **Tip**
>
> Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Una vez implementados los recursos del proyecto de ejemplo, haga lo siguiente.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.
    4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

4. Ve a tu grupo CloudWatch de registros e inspecciona los registros. El nombre del grupo de registros tendrá el siguiente aspecto: ExpressLogGroup -wjalRxUtnFemi.

Código de la máquina de estado de ejemplo del elemento principal (flujos de trabajo estándar)

La máquina de estado de este proyecto de ejemplo se integra con los flujos de trabajo rápidos de Amazon SQS, Amazon SNS y Step Functions.

Navegue por esta máquina de estado de ejemplo para ver cómo Step Functions procesa las entradas de Amazon SQS y Amazon SNS y, a continuación, utiliza una máquina de estado de flujos de trabajo rápidos anidados para actualizar los sistemas de backend.

Para obtener más información acerca de cómo AWS Step Functions puede controlar otros servicios, consulte. AWS Uso AWS Step Functions con otros servicios

```
{
    "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
```

```
        }
    }
}
```

Ejemplo de rol de IAM para la máquina de estado principal

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

Política de Amazon SNS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Política de Amazon SQS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:123456789012:Checkpoint-SQSQueue-
je7MtGbClwBF",
            "Effect": "Allow"
        }
    ]
```

```
}
```

Política de ejecución de estados:

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

Código de la máquina de estado de ejemplo de la máquina de estado anidada (flujos de trabajo rápidos)

La máquina de estado de este proyecto de ejemplo actualiza la información de backend cuando la llama la máquina de estado principal.

Navegue por esta máquina de estado de ejemplo para ver cómo Step Functions actualiza los diferentes componentes de los sistemas de backend de comercio electrónico simulados.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.


```
{
  "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
    }
  }
}
```

Ejemplo de rol de IAM de la máquina de estado secundaria

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
            ],
            "Effect": "Allow"
        }
    ]
}
```

La siguiente política garantiza que haya permisos suficientes para los CloudWatch registros.

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

Para obtener información acerca de cómo configurar IAM cuando se utiliza Step Functions con otros servicios de AWS , consulte Políticas de IAM para servicios integrados.

Creación de un AWS CodeBuild proyecto (CodeBuild, Amazon SNS)

En este proyecto de ejemplo se muestra cómo AWS Step Functions crear un AWS CodeBuild proyecto, ejecutar pruebas y, a continuación, enviar una notificación de Amazon SNS.

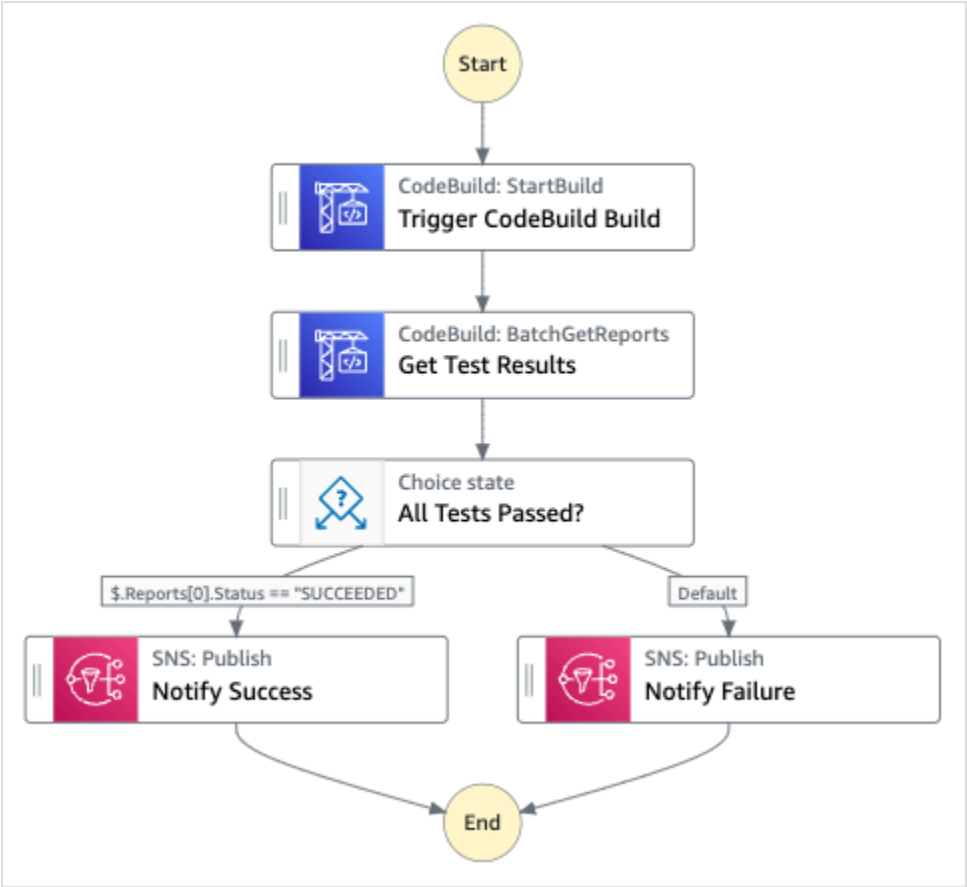
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Start a CodeBuild build`** en el cuadro de búsqueda y, a continuación, seleccione Iniciar una CodeBuild compilación a partir de los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Y AWS CodeBuild construye
   - Un tema de Amazon SNS
   - ¿Una máquina AWS Step Functions estatal
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   La siguiente imagen muestra el gráfico del flujo de trabajo del proyecto de ejemplo Start a CodeBuild build:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con CodeBuild Amazon SNS.

Examine este ejemplo de máquina de estados para ver cómo Step Functions utiliza una máquina de estados para crear un CodeBuild proyecto y, a continuación, envía un tema de Amazon SNS con un mensaje sobre si el trabajo se realizó correctamente o no.

Para obtener más información sobre cómo Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Para obtener información acerca de cómo configurar IAM cuando se utiliza Step Functions con otros servicios de AWS , consulte Políticas de IAM para servicios integrados.

Preprocesamiento de datos y entrenamiento de un modelo de machine learning

Este proyecto de ejemplo demuestra cómo usar SageMaker y AWS Step Functions preprocesar datos y cómo entrenar un modelo de aprendizaje automático.

En este proyecto, Step Functions utiliza una función de Lambda para iniciar un bucket de Amazon S3 con un conjunto de datos de prueba y un script de Python para el procesamiento de datos. A continuación, entrena un modelo de aprendizaje automático y realiza una transformación por lotes mediante la integración SageMaker de servicios.

Para obtener más información sobre SageMaker las integraciones de los servicios Step Functions, consulte lo siguiente:

- Uso AWS Step Functions con otros servicios

- Administre SageMaker con Step Functions

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y la capa gratuita, consulta SageMaker los precios.

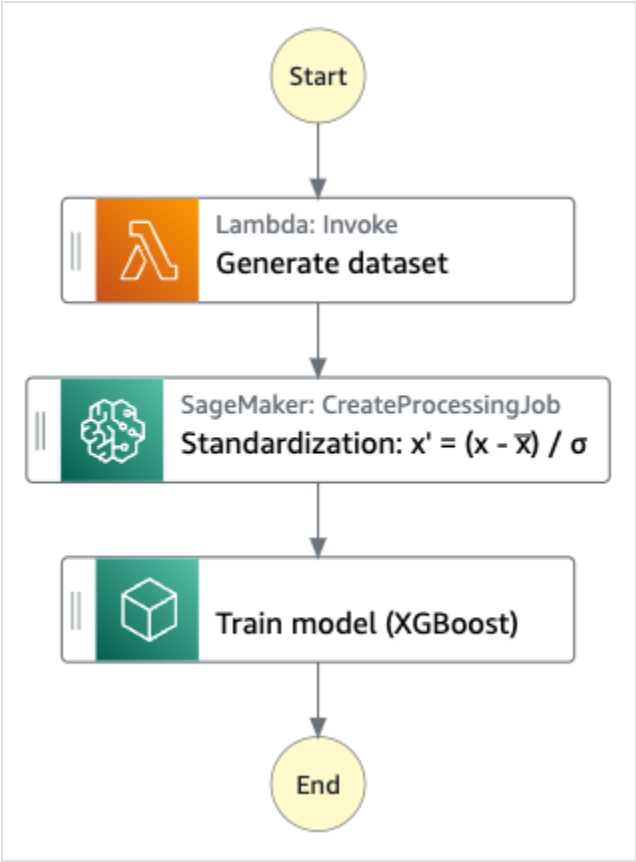
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Preprocess data and train a machine learning model`** en el cuadro de búsqueda y, a continuación, seleccione Preprocesamiento de datos y entrenamiento de un modelo de machine learning en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - ¿Una AWS Lambda función
   - Un bucket de Amazon S3.
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico de flujo del trabajo del proyecto de muestra Procesamiento previo de los datos y entrenamiento de un modelo de aprendizaje automático:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con esos recursos SageMaker y AWS Lambda los transfiere directamente a ellos, y utiliza un bucket de Amazon S3 para la fuente y la salida de los datos de entrenamiento.

Examine este ejemplo de máquina de estados para ver cómo Step Functions controla Lambda y. SageMaker

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

La siguiente política permite a la función de Lambda propagar los datos de muestra en el bucket de Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplos de orquestación de Lambda

Este proyecto de ejemplo muestra cómo integrar AWS Lambda funciones en máquinas de estados de Step Functions.

En este proyecto, Step Functions utiliza funciones de Lambda para consultar el precio de una acción y determinar una recomendación de compra o venta. A continuación, el usuario recibe esta recomendación y puede decidir si compra o vende las acciones. El resultado de la operación se devuelve mediante un tema de SNS.

Para obtener más información acerca de las integraciones de servicios de Step Functions, consulte los temas siguientes.

- [Uso AWS Step Functions con otros servicios](#)
- Políticas de IAM para:
  - [Políticas de IAM para AWS Lambda](#)
  - [Políticas de IAM para Amazon SQS](#)
  - [Políticas de IAM para Amazon SNS](#)

> **Note**
>
> Este proyecto de muestra puede generar cargos.

Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y el nivel gratuito, consulta los precios.

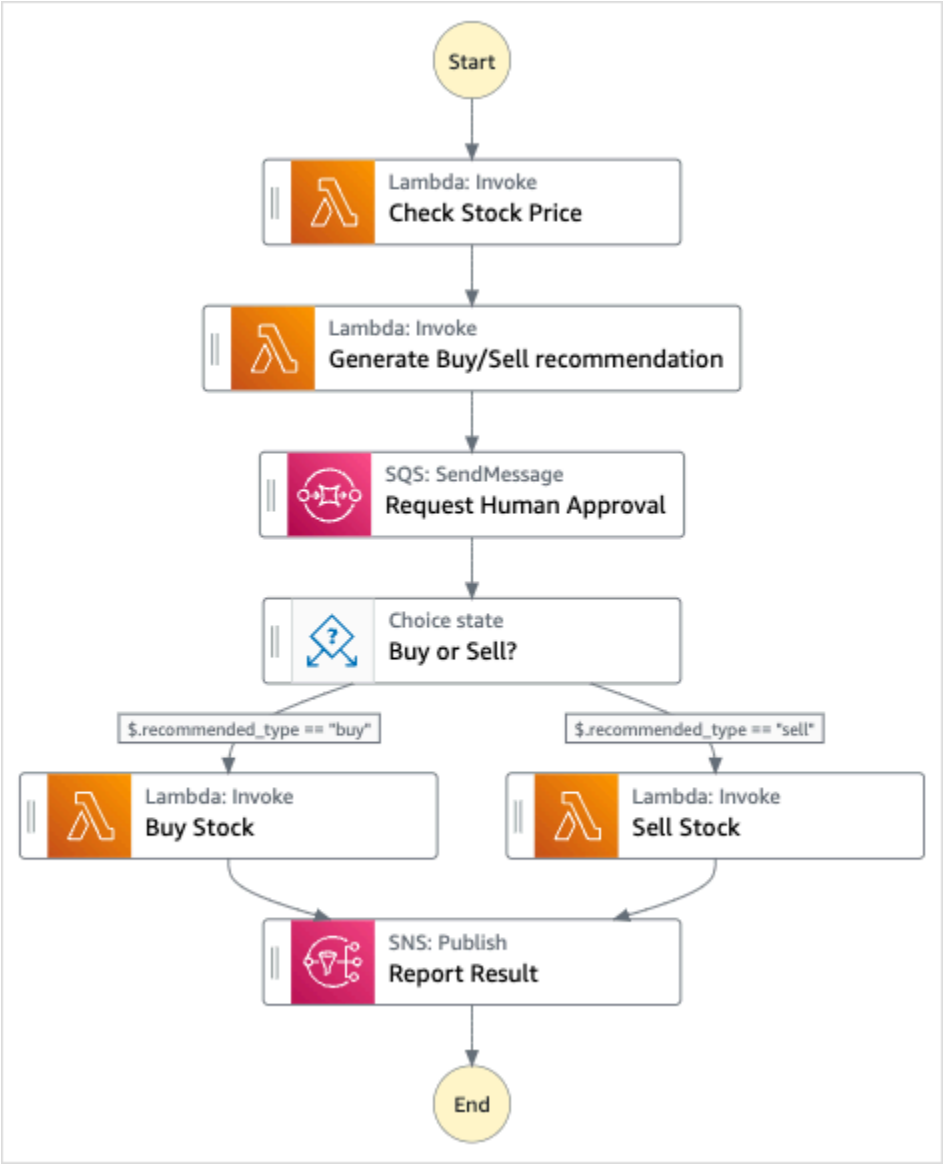
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Orchestrate Lambda functions`** en el cuadro de búsqueda y, a continuación, seleccione Organizar funciones de Lambda en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Cinco funciones de Lambda
   - Una cola de Amazon Simple Queue Service
   - Un tema de Amazon Simple Notification Service
   - Una máquina de estado de AWS Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Orquestar funciones de Lambda:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la

consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

Una vez aprovisionados e implementados todos los recursos, aparece el cuadro de diálogo Iniciar ejecución.

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.
    4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Acerca de la máquina de estado y su ejecución

La máquina de estados de este proyecto de ejemplo se integra AWS Lambda pasando los parámetros directamente a esos recursos, utiliza una cola de Amazon SQS para gestionar la solicitud de aprobación humana y utiliza un tema de Amazon SNS para devolver los resultados de la consulta.

Una ejecución de Step Functions recibe un texto JSON como entrada y transfiere dicha entrada al primer estado en el flujo de trabajo. Los estados individuales reciben datos de JSON como entrada y normalmente pasan datos de JSON como salida al siguiente estado. En este proyecto de muestra, la salida de cada paso se transfiere como entrada al siguiente paso del flujo de trabajo. Por ejemplo, el paso Generar recomendación de compra/venta recibe como la salida del paso Consultar el precio de una acción como entrada. Además, la salida del paso Generar recomendación de compra/venta se transfiere como entrada al siguiente paso, Solicitar aprobación humana, que imita un paso de aprobación humana.

> **Note**
>
> Para ver el resultado devuelto por un paso y la entrada transferida a un paso, abra la página Detalles de la ejecución de su flujo de trabajo. En la sección Detalles del paso, visualice la entrada y la salida de cada paso que seleccione en el Modo de visualización.

Para implementar un paso de aprobación humana, normalmente se pausa la ejecución del flujo de trabajo hasta que se devuelva un token de tarea. En este proyecto de muestra, se transfiere un mensaje a una cola de Amazon SQS, que se utiliza como activador de la función de Lambda definida para gestionar la funcionalidad de devolución de llamadas. El mensaje contiene un token de tarea y la salida devuelta en el paso anterior. La función de Lambda se invoca con la carga del mensaje. La ejecución del flujo de trabajo se detiene hasta que recibe el token de tarea con una llamada a la API `SendTaskSuccess`. Para obtener más información acerca de los tokens de tarea, consulte Cómo esperar una devolución de llamada con el token de tarea.

El siguiente código de la función `StepFunctionsSample-HelloLambda-ApproveSqsLambda` muestra cómo se define para aprobar automáticamente cualquier tarea enviada por la cola de Amazon SQS a través de la máquina de estado de Step Functions.

Código de función de Lambda de muestra para gestionar la funcionalidad de devolución de llamada y devolver el token de tarea

```
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);
      });
  }
};
```

Examine este ejemplo de máquina de estado para ver cómo Step Functions controla Lambda y Amazon SQS.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
    "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
        }
    }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplos de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte Políticas de IAM para servicios integrados.

Iniciar una consulta de Athena

En este proyecto de muestra, que se basa en flujos de trabajo estándar, se ilustra cómo usar Step Functions y Amazon Athena para iniciar una consulta de Athena y enviar una notificación con los resultados de la consulta.

En este proyecto, Step Functions utiliza funciones Lambda y un AWS Glue rastreador para generar un conjunto de datos de ejemplo. A continuación, realiza una consulta mediante la integración del servicio de Athena y devuelve los resultados mediante un tema de SNS.

Para obtener más información acerca de de Athena y de las integraciones de servicios de Step Functions, consulte los temas siguientes.

- Uso AWS Step Functions con otros servicios
- Llamar a Athena con Step Functions

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costes y la capa gratuita, consulta los precios de Athena.

Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Start an Athena query`** en el cuadro de búsqueda y, a continuación, seleccione Iniciar una consulta de Athena en de los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de

partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

En este proyecto de muestra se implementan los siguientes recursos:

- Una consulta de Amazon Athena
- Una Rastreador de AWS Glue
- Un tema de Amazon SNS
- Una máquina de estado de AWS Step Functions
- Roles de AWS Identity and Access Management (IAM) relacionados

En la siguiente imagen se ilustra el gráfico de flujo del trabajo del proyecto de muestra Iniciar una consulta de Athena:

5. Elija Utilizar plantilla para continuar con la selección.

6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

     > **Tip**
     >
     > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

     Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

     El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

     Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con Athena y pasa AWS Lambda los parámetros directamente a esos recursos, y utiliza un tema de SNS para devolver los resultados de la consulta.

Examine este ejemplo de máquina de estado para ver cómo Step Functions controla Lambda y Athena.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejecutar varias consultas (Amazon Athena, Amazon SNS)

En este proyecto de muestra se ilustra cómo ejecutar consultas de Athena de forma consecutiva y luego en paralelo, tratar los errores y, a continuación, enviar una notificación de Amazon SNS en función de si las consultas se realizan correctamente o no.

En este proyecto, Step Functions utiliza una máquina de estado para ejecutar consultas de Athena de forma sincrónica. Una vez devueltos los resultados de la consulta, introduzca el estado Parallel con dos consultas de Athena ejecutándose en paralelo. A continuación, espera a que el trabajo se realice correcta o incorrectamente y envía un tema de Amazon SNS con un mensaje que informa sobre si el trabajo ha finalizado correctamente o con errores.

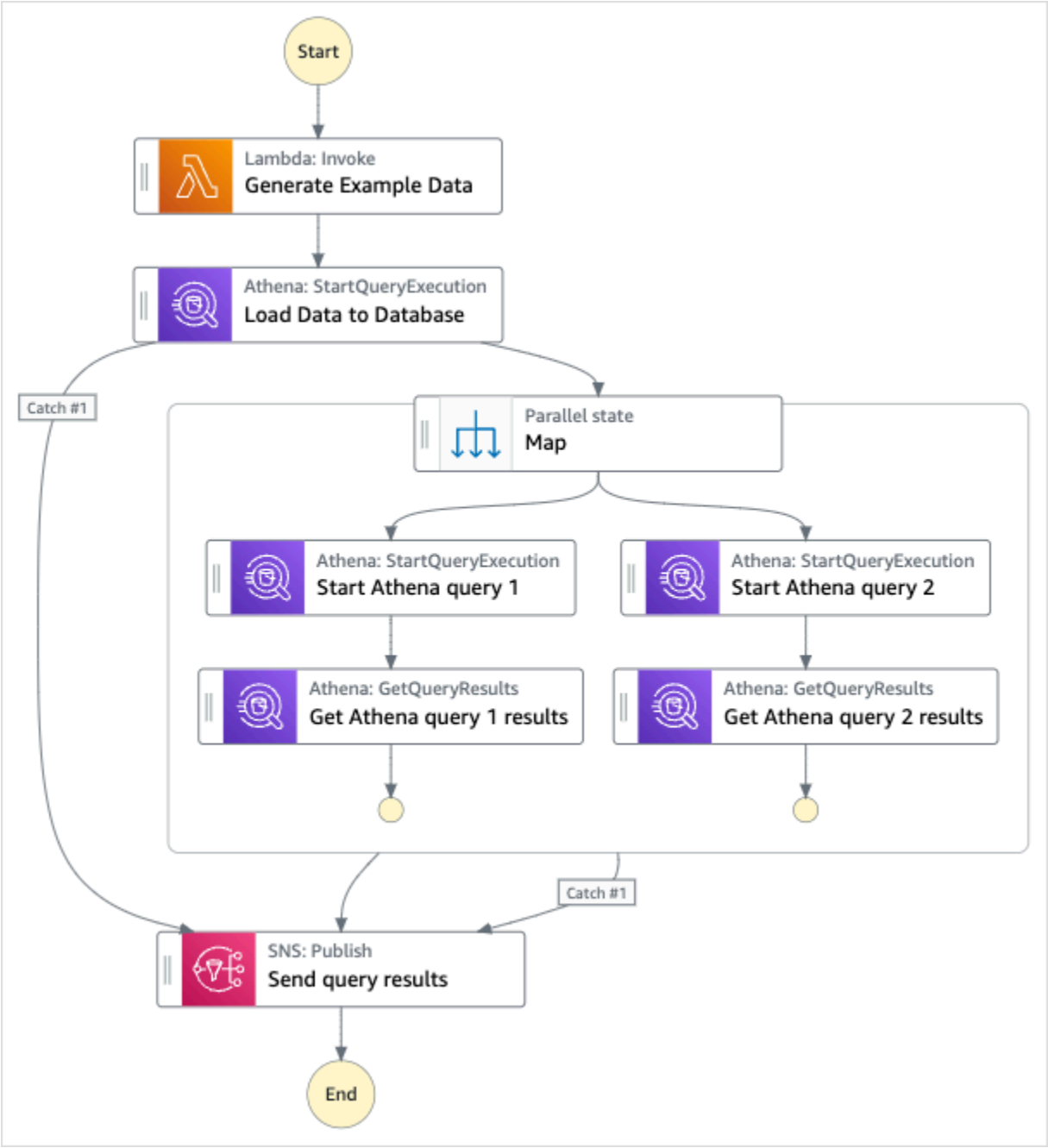
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Execute multiple queries`** en el cuadro de búsqueda y, a continuación, seleccione Ejecutar varias consultas en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Consultas de Amazon Athena
   - Un tema de Amazon SNS
   - Una máquina de estado de AWS Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Ejecutar varias consultas:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al

Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

> **Tip**
>
> Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

    3. Seleccione Iniciar ejecución.
    4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

        Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con Amazon Athena y Amazon SNS pasando parámetros directamente a esos recursos.

Examine este ejemplo de máquina de estado para ver cómo Step Functions controla Amazon Athena y Amazon SNS conectándose al nombre de recurso de Amazon (ARN) del campo `Resource` y pasando `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
    "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
      }
    }
  }
```

Ejemplos de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

SNSPublish

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Consulte conjuntos de datos de gran tamaño (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)

En este proyecto de ejemplo se muestra cómo ingerir un conjunto de datos de gran tamaño en Amazon S3 y particionarlo mediante AWS Glue rastreadores y, a continuación, ejecutar consultas de Amazon Athena en esa partición.

En este proyecto, la máquina de estados Step Functions invoca un AWS Glue rastreador que particiona un conjunto de datos grande en Amazon S3. Una vez que el AWS Glue rastreador devuelve un mensaje de éxito, el flujo de trabajo ejecuta las consultas de Athena en esa partición. Una vez que la ejecución de la consulta se haya completado correctamente, se envía una notificación de Amazon SNS a un tema de Amazon SNS.

Paso 1: Crear la máquina de estado y aprovisionar recursos

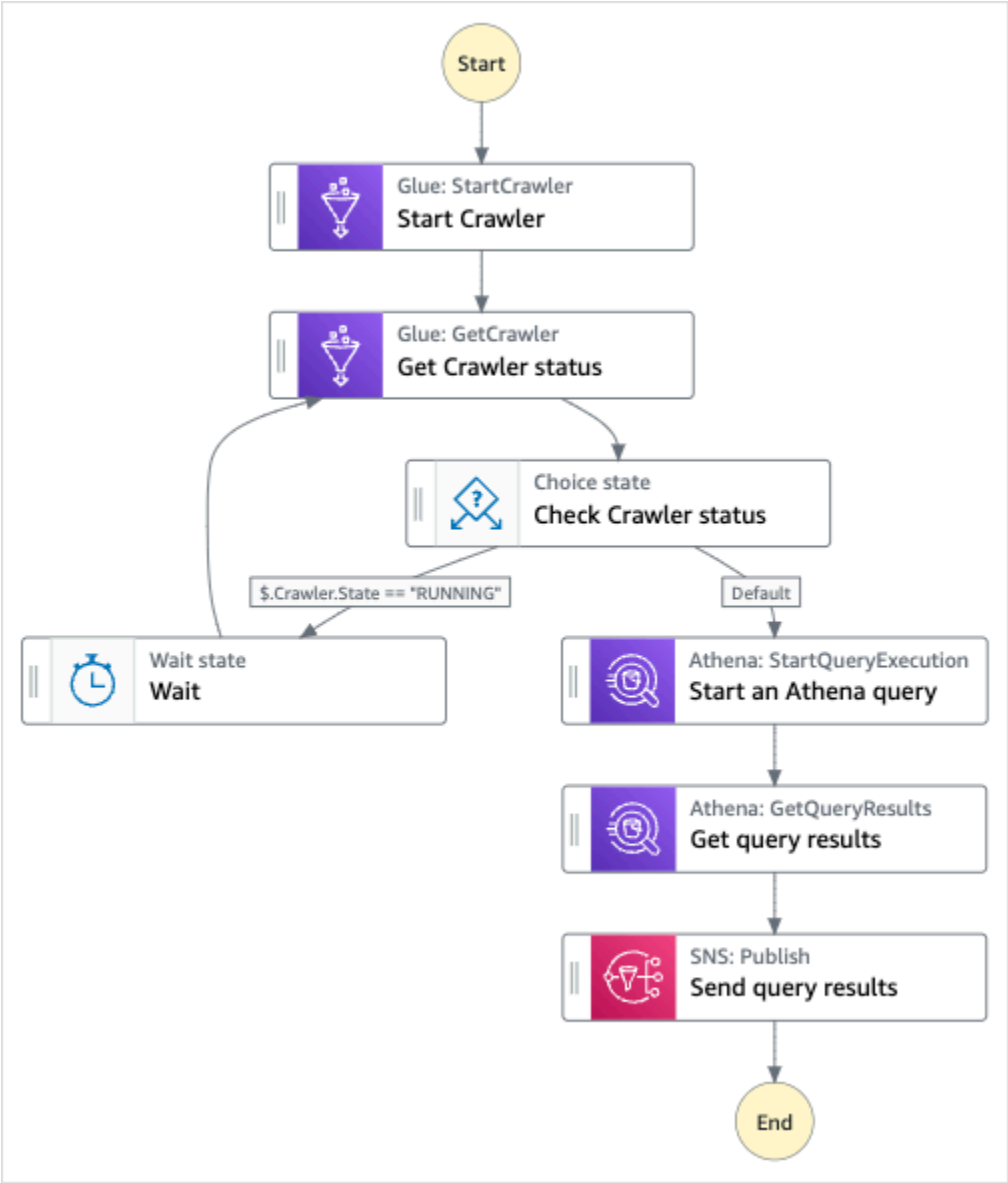
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Query large datasets`** en el cuadro de búsqueda y, a continuación, seleccione Consultar conjuntos de datos de gran tamaño en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.

4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Un bucket de Amazon S3
   - Una Rastreador de AWS Glue
   - Un tema de Amazon SNS
   - Una máquina de estado de AWS Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

   En la siguiente imagen se ilustra el gráfico de flujo del trabajo del proyecto de muestra Consultar conjuntos de datos de gran tamaño:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la

consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

> **Tip**
>
> Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Se pueden aplicar cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.

2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.
    3. Seleccione Iniciar ejecución.
    4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

        Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con Amazon S3 AWS Glue, Amazon Athena y Amazon SNS al pasar los parámetros directamente a esos recursos.

Explore este ejemplo de máquina de estados para ver cómo Step Functions controla Amazon S3 AWS Glue, Amazon Athena y Amazon SNS conectándose al nombre de recurso de Amazon (ARN) en `Resource` el campo y pasándolo `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
    "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
      }
    }
  }
```

Ejemplos de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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": [
                "*"
            ]
        }
    ]
}
```

SNSPublish

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Mantener los datos actualizados (Amazon Athena, Amazon S3,) AWS Glue

Este proyecto de ejemplo muestra cómo consultar una tabla de destino para obtener datos actuales con AWS Glue Catalog y, a continuación, actualizarla con nuevos datos de otras fuentes mediante Amazon Athena.

En este proyecto, la máquina de estados Step Functions llama a AWS Glue Catalog para verificar si existe una tabla de destino en un bucket de Amazon S3. Si no encuentra ninguna tabla, se creará una nueva. A continuación, Step Functions ejecuta una consulta de Athena para añadir filas a la tabla de destino desde un origen de datos diferente: primero consulta la tabla de destino para obtener la fecha más reciente y, posteriormente, consulta la tabla de origen para obtener datos más recientes e insertarlos en la tabla de destino.

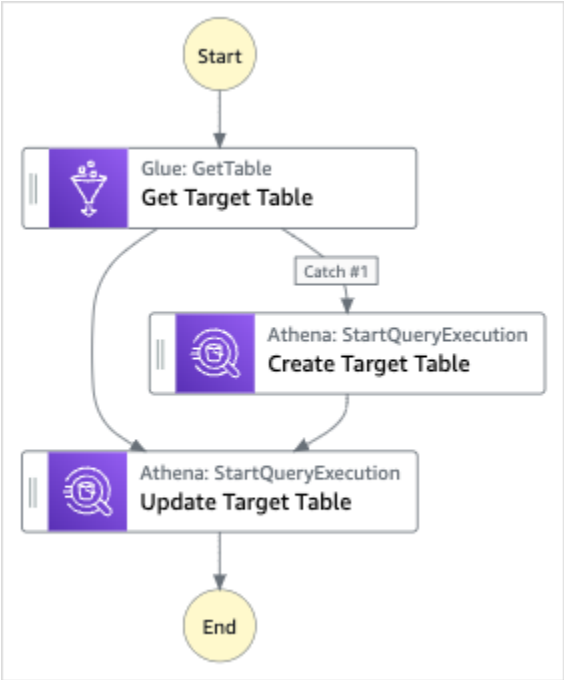
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Keep data up to date`** en el cuadro de búsqueda y, a continuación, seleccione Mantener los datos actualizados en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Un bucket de Amazon S3
   - Consultas de Amazon Athena
   - Una llamada de AWS Glue Data Catalog
   - Una máquina de estado de AWS Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Mantener los datos actualizados:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

   - Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

> **Tip**
>
> Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

    > **Note**
    >
    > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede

realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.
3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con Amazon S3 y Amazon Athena al pasar los parámetros directamente a esos recursos. AWS Glue

Explore este ejemplo de máquina de estados para ver cómo Step Functions controla Amazon S3 y Amazon Athena conectándose al nombre de recurso de Amazon (ARN) en el `Resource` campo y pasándolo `Parameters` a la API del servicio. AWS Glue

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
    "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
      }
    }
  }
```

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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": [
                "*"
            ]
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Administración de un clúster de Amazon EKS

En este proyecto de muestra se explica cómo utilizar Step Functions y Amazon Elastic Kubernetes Service para crear un clúster de Amazon EKS con un grupo de nodos, ejecutar un trabajo en Amazon EKS y, a continuación, examinar el resultado. Cuando termina, se eliminan los grupos de nodos y el clúster de Amazon EKS.

Para obtener más información acerca de Step Functions y de sus integraciones de servicios, consulte lo siguiente:

- Uso AWS Step Functions con otros servicios
- Llamar a Amazon EKS con Step Functions

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costes y la capa gratuita, consulta los precios de Amazon EKS.

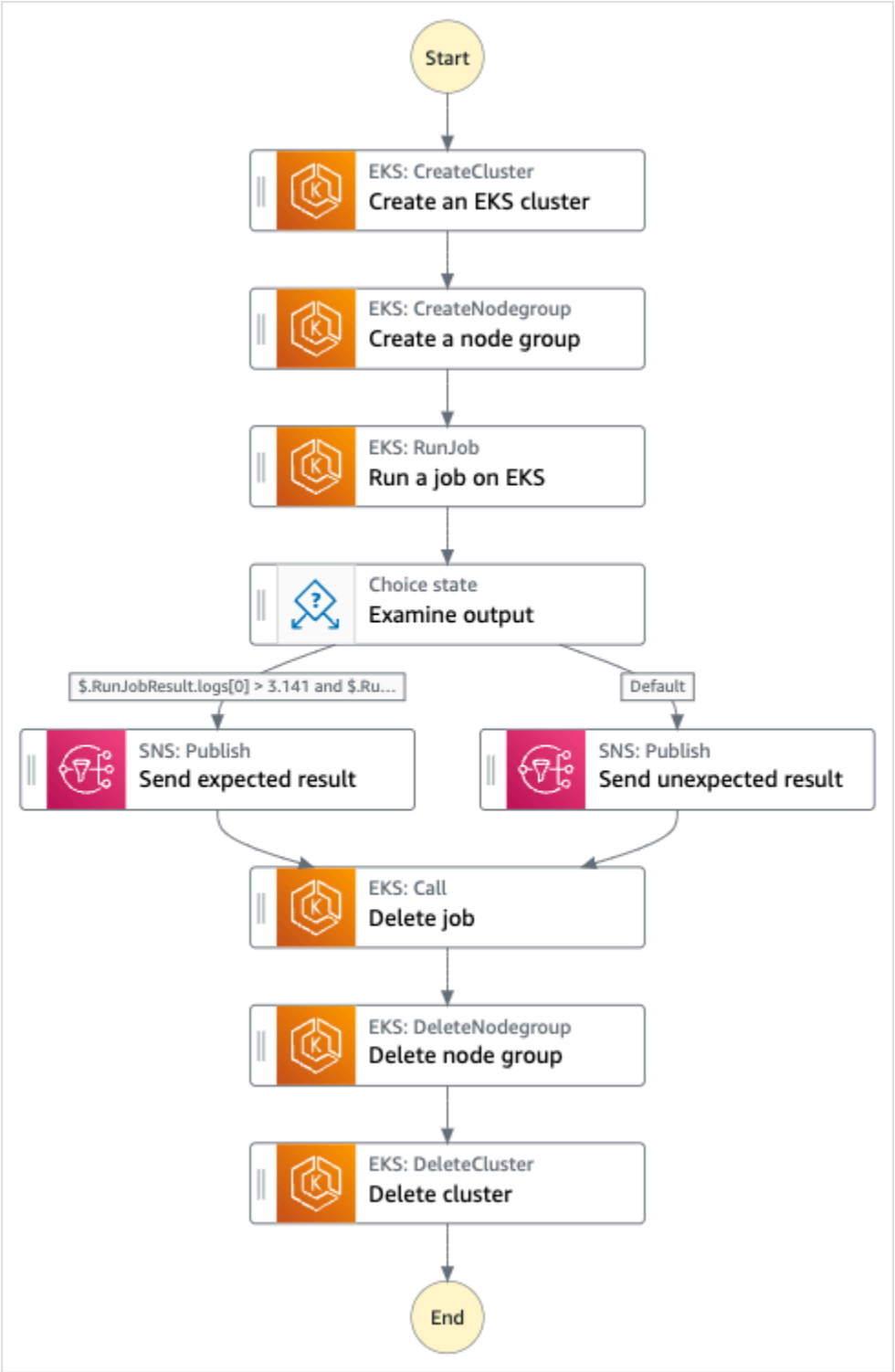
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Manage an EKS cluster`** en el cuadro de búsqueda y, a continuación, seleccione Administración de un clúster de EKS en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Un clúster Amazon Elastic Kubernetes Service
   - Un tema de Amazon SNS
   - Una máquina de estado de AWS Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Administración de un clúster de EKS:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

  > **Important**
  >
  > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Se pueden aplicar cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con Amazon EKS mediante la creación de un clúster y un grupo de nodos de Amazon EKS, y utiliza un tema de SNS para devolver los resultados.

Examine este ejemplo de máquina de estado para ver cómo Step Functions administra los clústeres y grupos de nodos de Amazon EKS.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

```
{
    "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"
```

```
            ]
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte Políticas de IAM para servicios integrados.

Hacer una llamada a API Gateway

En este proyecto de muestra se explica cómo utilizar Step Functions para hacer una llamada a API Gateway y se comprueba si la llamada se realizó correctamente.

Para obtener más información acerca de API Gateway y de las integraciones de servicios de Step Functions, consulte los temas siguientes.

- Uso AWS Step Functions con otros servicios
- Llamar a API Gateway con Step Functions

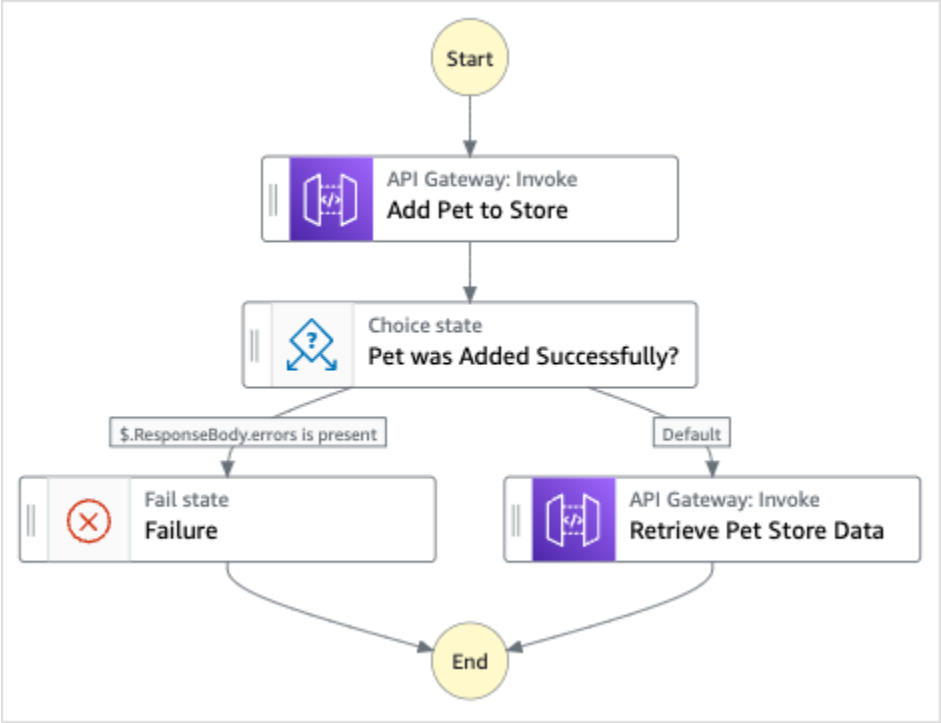
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Make a call to API Gateway`** en el cuadro de búsqueda y, a continuación, seleccione Hacer una llamada a API Gateway en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una API de REST de Amazon API Gateway a la que llama la máquina de estados.
   - Una máquina de estado de AWS Step Functions
   - Roles de AWS Identity and Access Management (IAM) relacionados

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Hacer una llamada a API Gateway:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con API Gateway llamando a la API de REST de API Gateway y pasando los parámetros necesarios.

Examine este ejemplo de máquina de estado para ver cómo interactúa Step Functions con API Gateway.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Llamar a un microservicio que se ejecute en Fargate mediante la integración de API Gateway

En este proyecto de ejemplo se muestra cómo utilizar Step Functions para realizar una llamada a API Gateway con el fin de interactuar con un servicio activo AWS Fargate y también para comprobar si la llamada se ha realizado correctamente.

Para obtener más información acerca de API Gateway y de las integraciones de servicios de Step Functions, consulte los temas siguientes.

- [Uso AWS Step Functions con otros servicios](#)
- [Llamar a API Gateway con Step Functions](#)

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y la capa gratuita, consulta [los precios](#).

Paso 1: Crear la máquina de estado y aprovisionar recursos

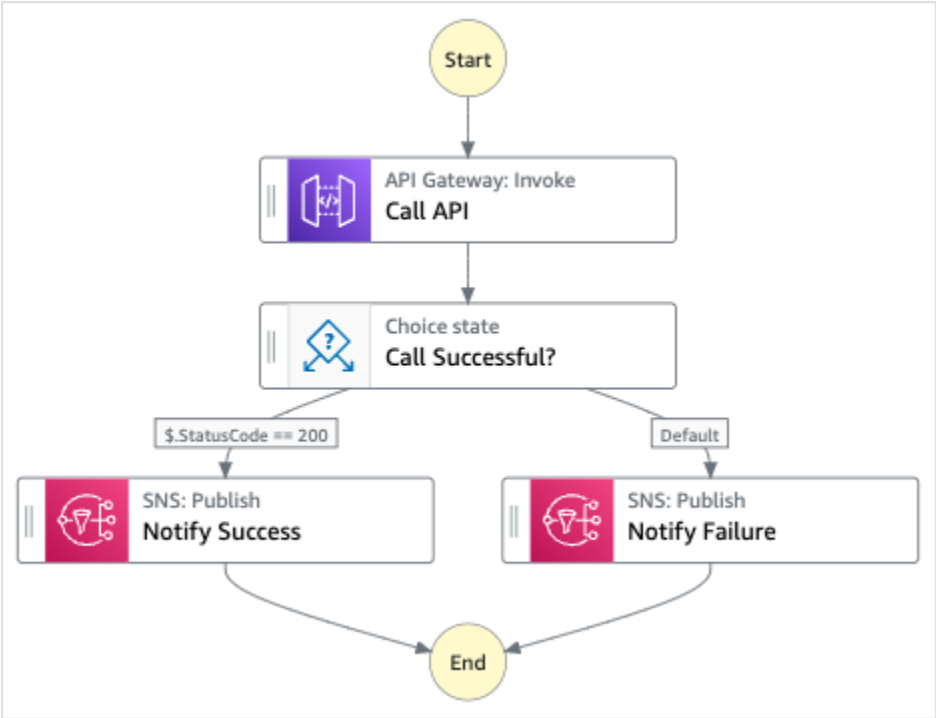
1. Abra la [consola de Step Functions](#) y seleccione Crear máquina de estado.
2. Escriba **`Call a microservice with API Gateway`** en el cuadro de búsqueda y, a continuación, seleccione Llamar a un microservicio con API Gateway en los resultados de búsqueda que aparecen.

3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una API HTTP de Amazon API Gateway a la que llama la máquina de estado.
   - Un enlace de Amazon API Gateway Amazon VPC.
   - Una Amazon Virtual Private Cloud.
   - Una Application Load Balancer.
   - Un clúster de Fargate.
   - Un tema de Amazon SNS
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)
   - Se requieren varios servicios adicionales para que estos recursos puedan funcionar juntos.

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Llamar a un microservicio con API Gateway:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con API Gateway al llamar a una API HTTP de API Gateway que está conectada a un servicio en Fargate. Se aloja en una subred privada y se accede a ella mediante un equilibrador de carga de aplicación privado.

Examine este ejemplo de máquina de estado para ver cómo interactúa Step Functions con API Gateway y devuelve resultados.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulte[Uso AWS Step Functions con otros servicios](#).

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte Políticas de IAM para servicios integrados.

Envía un evento personalizado a EventBridge

Este proyecto de ejemplo muestra cómo usar Step Functions para enviar un evento personalizado a un bus de eventos que cumpla con una regla con varios objetivos (Amazon EventBridge AWS Lambda, Amazon Simple Notification Service, Amazon Simple Queue Service).

Para obtener más información acerca de Step Functions y de sus integraciones de servicios, consulte lo siguiente:

- Uso AWS Step Functions con otros servicios
- Llama EventBridge con Step Functions

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y la capa gratuita, consulta EventBridge los precios.

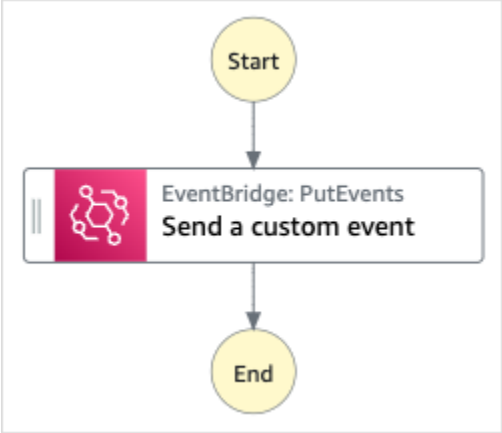
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Send a custom event to EventBridge`** en el cuadro de búsqueda y, a continuación, seleccione Enviar un evento personalizado a EventBridge en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

- Un evento de Amazon EventBridge
- Un tema de Amazon SNS
- Una cola de Amazon SQS
- Una función de Lambda
- ¿Una máquina de AWS Step Functions estados
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Enviar un evento personalizado a EventBridge:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra EventBridge enviando un evento personalizado a un bus de EventBridge eventos. El evento enviado al bus de eventos coincide con una EventBridge regla que activa una función Lambda que envía mensajes a un tema de Amazon SNS y a una cola de Amazon SQS.

Examine este ejemplo de máquina de estados para ver cómo se gestiona Step Functions EventBridge.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
  "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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:1234567890:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Invocar flujos de trabajo rápidos sincrónicos

En este proyecto de muestra se ilustra cómo invocar flujos de trabajo rápidos sincrónicos a través de Amazon API Gateway para gestionar una base de datos de empleados.

En este proyecto, Step Functions utiliza puntos de conexión de API Gateway para iniciar los flujos de trabajo rápidos sincrónicos de Step Functions. A continuación, se utiliza DynamoDB para buscar, añadir y eliminar empleados en una base de datos de empleados.

Para obtener más información acerca de los flujos de trabajo rápidos sincrónicos de Step Functions, consulte Flujos de trabajo rápidos síncronos y asíncronos.

> **Note**
>
> Este proyecto de muestra puede generar cargos.

Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costes y la capa gratuita, consulta los precios de Step Functions.

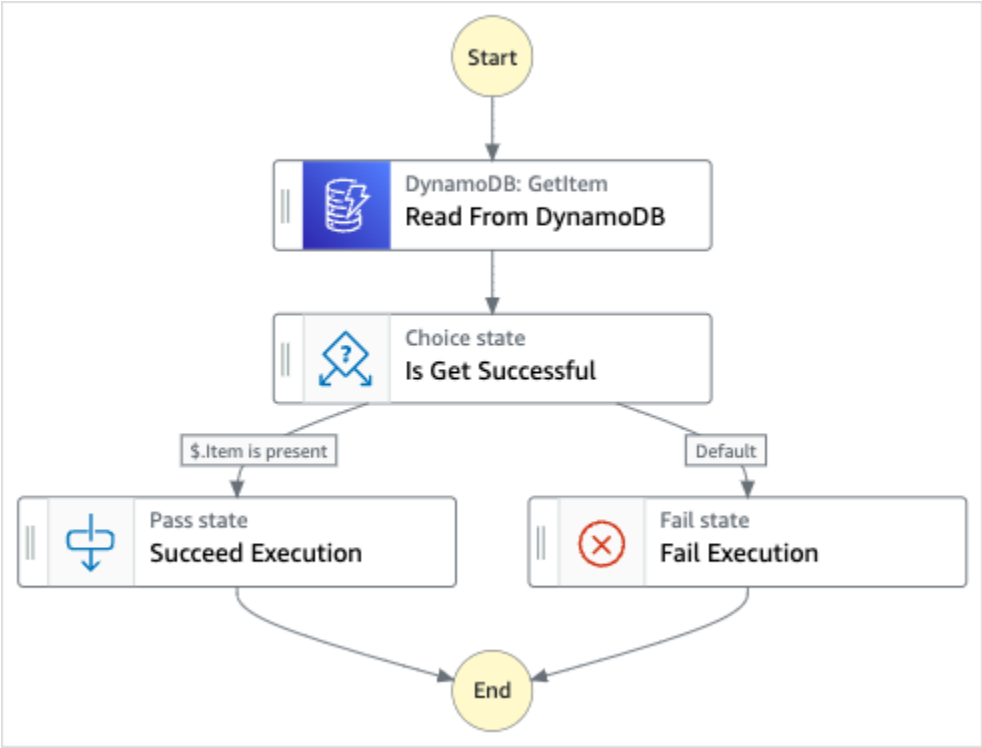
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Invoke Synchronous Express Workflows through API Gateway`** en el cuadro de búsqueda y, a continuación, seleccione Invocar flujos de trabajo Synchronous Express a través de API Gateway en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una API de HTTPS de Amazon API Gateway a la que llama una máquina de estado.
   - Una tabla de Amazon DynamoDB.
   - Tres máquinas AWS Step Functions estatales.
   - Funciones relacionadas AWS Identity and Access Management (IAM).

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Invocar flujos de trabajo rápidos sincrónicos a través de API Gateway:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

     > **Important**
     >
     > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

> **Note**
>
> Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.
4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estado de este proyecto de muestra se integra con API Gateway y DynamoDB mediante API Gateway para invocar un flujo de trabajo rápido sincrónico, que posteriormente actualiza o lee la base de datos de empleados mediante DynamoDB.

Examine esta máquina de estado de ejemplo para ver cómo Step Functions lee datos de DynamoDB para recuperar la información de los empleados.

Para obtener más información sobre cómo invocar Step Functions mediante API Gateway, consulte lo siguiente.

- Llamar a API Gateway con Step Functions
- Cómo invocar una puerta de enlace privada en la Guía para desarrolladores de API Gateway.

Para obtener más información sobre cómo AWS Step Functions controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte[Políticas de IAM para servicios integrados](#).

Ejemplos de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Ejecutar flujos de trabajo de ETL/ELT con Amazon Redshift (Lambda, API de datos de Amazon Redshift)

En este proyecto de muestra se ilustra cómo utilizar Step Functions y la API de datos de Amazon Redshift para ejecutar un flujo de trabajo de ETL/ELT que carga datos en el almacén de datos de Amazon Redshift.

En este proyecto, Step Functions utiliza una AWS Lambda función y la API Amazon Redshift Data para crear los objetos de base de datos necesarios y generar un conjunto de datos de ejemplo. A continuación, ejecuta dos trabajos en paralelo que cargan tablas de dimensiones, seguidas de

una tabla de hechos. Cuando ambos trabajos de carga de dimensiones finalicen correctamente, Step Functions ejecuta el trabajo de carga de la tabla de hechos, ejecuta el trabajo de validación y, posteriormente, detiene el clúster de Amazon Redshift.

> **Note**
>
> Puede modificar la lógica de ETL para recibir datos de otros orígenes, como Amazon S3, que puede utilizar el comando COPY para copiar datos de Amazon S3 a una tabla de Amazon Redshift.

Para obtener más información acerca de las integraciones de servicios de Amazon Redshift y Step Functions, consulte los siguientes temas:

- Uso AWS Step Functions con otros servicios
- Uso de la API de datos de Amazon Redshift
- Servicio de API de datos de Amazon Redshift
- Creación de una máquina de estado de Step Functions que utilice Lambda
- Políticas de IAM para:
  - Políticas de IAM para AWS Lambda
  - Autorización del acceso a la API de datos de Amazon Redshift

> **Note**
>
> Este proyecto de muestra puede generar cargos.
> Para AWS los nuevos usuarios, hay disponible un nivel de uso gratuito. En esta capa, los servicios son gratuitos por debajo de determinado nivel de uso. Para obtener más información sobre AWS los costos y la capa gratuita, consulta AWS Step Functions los precios.

Paso 1: Crear la máquina de estado y aprovisionar recursos

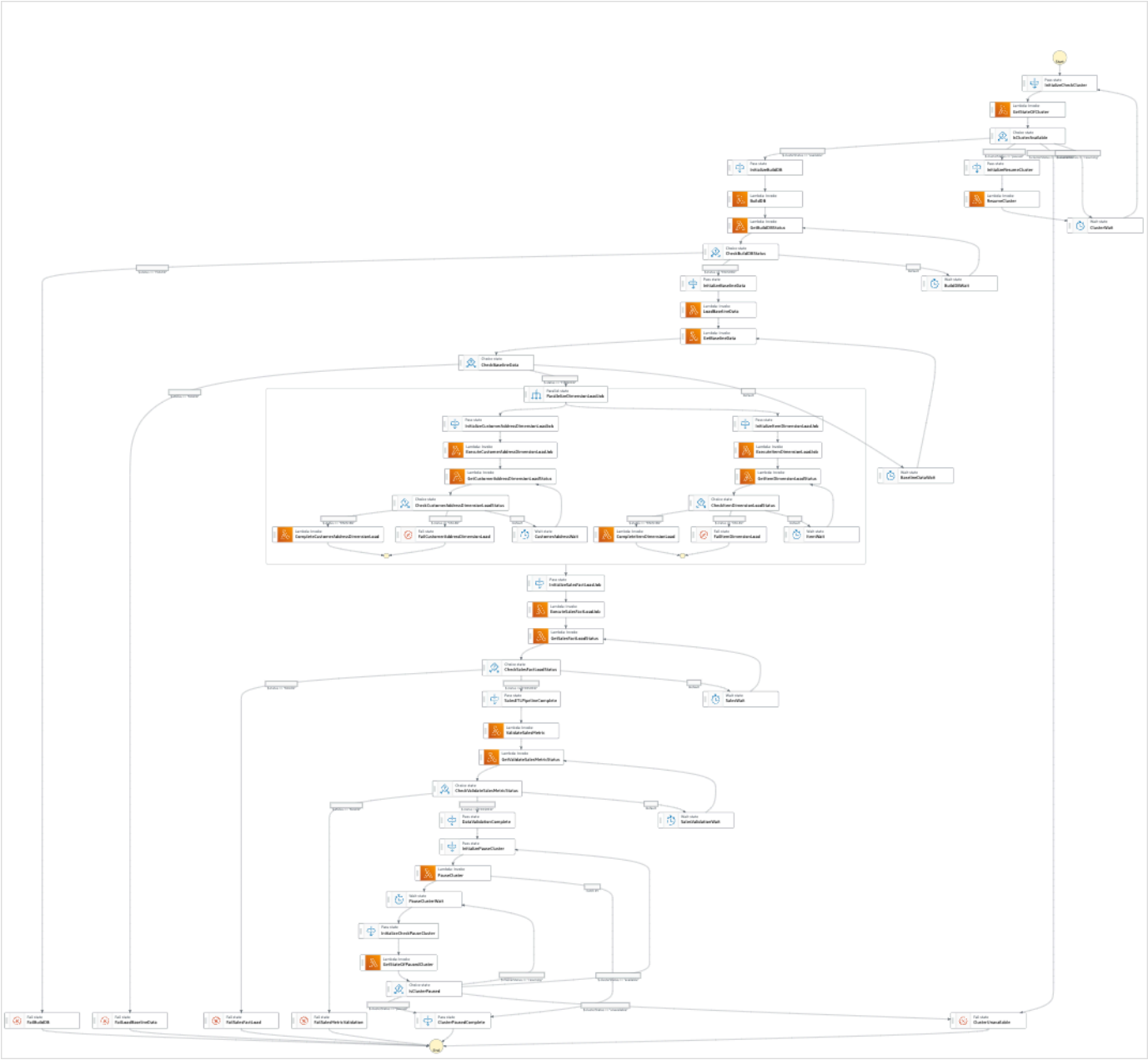
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`ETL job in Amazon Redshift`** en el cuadro de búsqueda y, a continuación, seleccione Trabajo de ETL en Amazon Redshift en los resultados de búsqueda que aparecen.

3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Un clúster Amazon Redshift
   - Dos funciones de Lambda
   - Un esquema de Amazon Redshift
   - Cinco tablas de Amazon Redshift
   - ¿Una máquina de AWS Step Functions estados
   - Funciones relacionadas AWS Identity and Access Management (IAM).

   En la siguiente imagen se ilustra el gráfico de flujo del trabajo del proyecto de muestra Trabajo de ETL en Amazon Redshift:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar

la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

        > **Note**
        >
        > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

    3. Seleccione Iniciar ejecución.
    4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

        Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente.

Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra pasando la lógica ETL InputPath directamente a esos recursos y ejecutándose de forma asíncrona AWS Lambda mediante la API de datos de Amazon Redshift.

Examine este ejemplo de máquina de estados para ver cómo Step Functions controla AWS Lambda y la API de datos de Amazon Redshift.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulteUso AWS Step Functions con otros servicios.

```
{
"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'),",
                    "('AAAAAAAANBEBAAAA','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
    }
  }
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Usar Step Functions y AWS Batch con control de errores

En este proyecto de muestra se ilustra cómo usar Step Functions para ejecutar un trabajo de AWS Batch mediante una máquina de estado con funciones de control de errores.

En este proyecto, Step Functions utiliza una máquina de estado para llamar al trabajo de AWS Batch de forma síncrona. A continuación, espera a que el trabajo se realice correcta o incorrectamente, lo reintenta y detecta errores cuando se produce un error en un trabajo, y envía un tema de Amazon SNS con un mensaje que informa sobre si el trabajo ha finalizado correctamente o con errores.

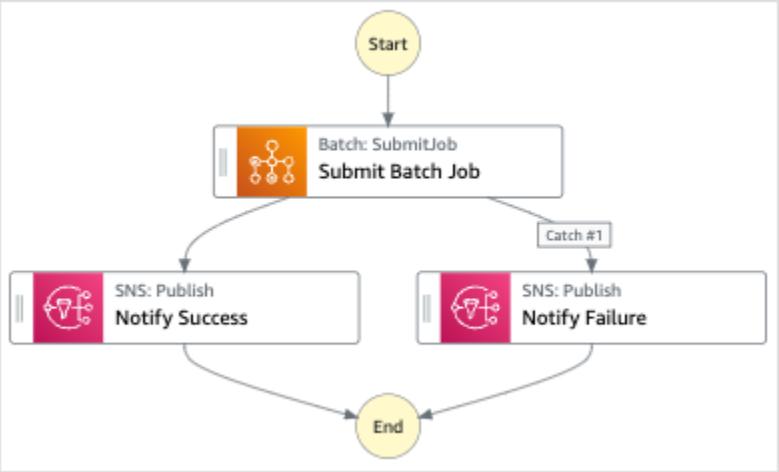
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Manage a batch job`** en el cuadro de búsqueda y, a continuación, seleccione Administración de un trabajo por lotes en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

En este proyecto de muestra se implementan los siguientes recursos:

- ¿Un AWS Batch trabajo
- Un tema de Amazon SNS
- ¿Una máquina AWS Step Functions estatal
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Administración de un trabajo por lotes:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con AWS Batch Amazon SNS al pasar los parámetros directamente a esos recursos.

Explore este ejemplo de máquina de estados para ver cómo Step Functions controla AWS Batch Amazon SNS conectándose al nombre de recurso de Amazon (ARN) en el `Resource` campo y pasándolo `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
  "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
    }
  }
}
```

Ejemplo de IAM

Esta política de ejemplo AWS Identity and Access Management (IAM) generada por el proyecto de muestra incluye los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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"
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

Haz un abanico de AWS Batch trabajo

Este proyecto de ejemplo demuestra cómo utilizar el Map estado de Step Functions para distribuir las AWS Batch tareas.

En este proyecto, Step Functions utiliza una máquina de estados para invocar una función Lambda para realizar un preprocesamiento sencillo y, a continuación, invoca varios AWS Batch trabajos en paralelo utilizando el estado. Map

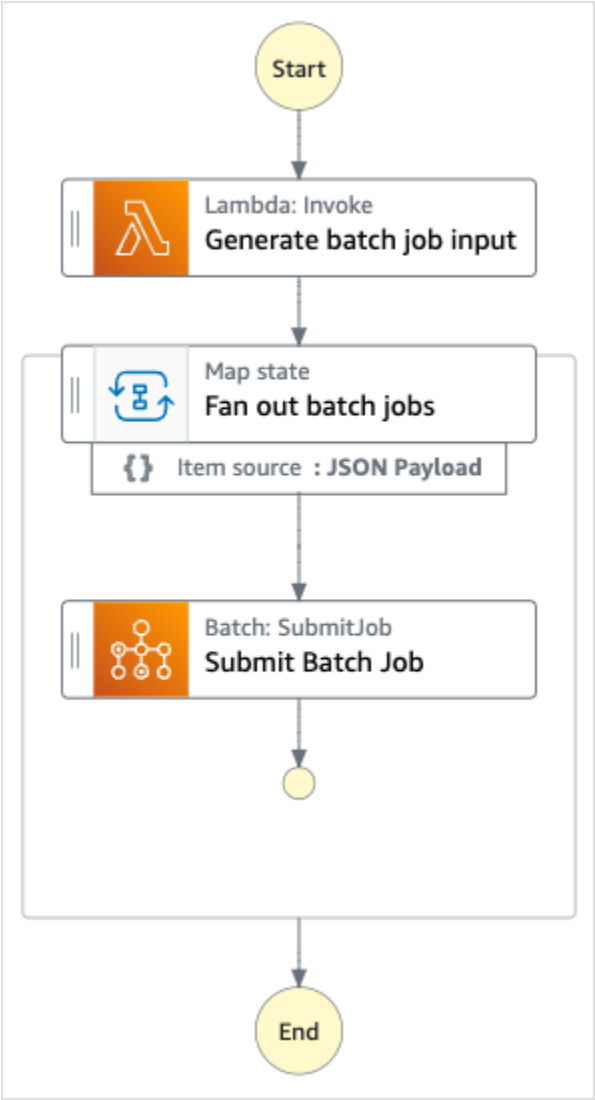
Paso 1: Crear la máquina de estado y aprovisionar recursos

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Fan out a batch job`** en el cuadro de búsqueda y, a continuación, seleccione Distribuir un trabajo por lotes en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

   En este proyecto de muestra se implementan los siguientes recursos:

   - Una función de Lambda
   - Una cola de trabajos de AWS Batch
   - Una máquina AWS Step Functions de estados
   - Funciones relacionadas AWS Identity and Access Management (IAM)

   En la siguiente imagen se ilustra el gráfico del flujo de trabajo del proyecto de muestra Distribuir un trabajo por lotes:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

  > **Important**
  >
  > Se pueden aplicar cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

   > **Note**
   >
   > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

   Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

   > **Note**
   >
   > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

3. Seleccione Iniciar ejecución.

4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

   Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con AWS Batch Amazon SNS pasando los parámetros directamente a esos recursos.

Explore este ejemplo de máquina de estados para ver cómo Step Functions controla AWS Batch Amazon SNS conectándose al nombre de recurso de Amazon (ARN) en el `Resource` campo y pasándolo `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

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

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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"
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consultePolíticas de IAM para servicios integrados.

AWS Batch con Lambda

En este proyecto de ejemplo se muestra cómo utilizar Step Functions para preprocesar datos con AWS Lambda funciones y, a continuación, organizar tareas. AWS Batch

En este proyecto, Step Functions utiliza una máquina de estado para invocar una función de Lambda y realizar un preprocesamiento sencillo antes de que se envíe un trabajo de AWS Batch . Se pueden invocar varios trabajos en función del resultado o éxito del anterior.

Paso 1: Crear la máquina de estado y aprovisionar recursos

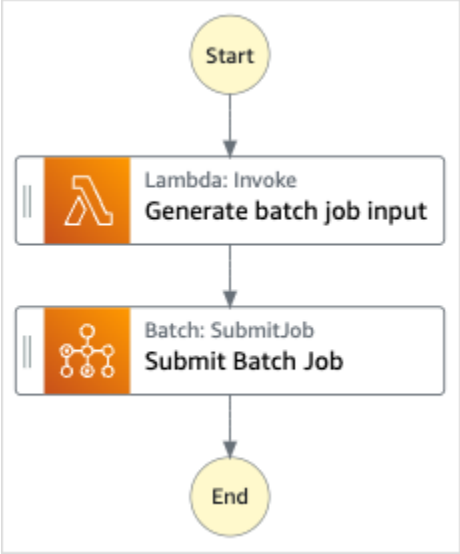
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`Batch job with Lambda`** en el cuadro de búsqueda y, a continuación, seleccione Trabajo por lotes con Lambda en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de

muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

En este proyecto de muestra se implementan los siguientes recursos:

- Una función de Lambda
- Un trabajo de AWS Batch
- Una máquina AWS Step Functions de estados
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo para el proyecto de muestra Trabajo por lotes con Lambda:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

   - Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

     En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la

consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

> **Important**
>
> No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

> **Tip**
>
> Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Se pueden aplicar cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.

2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

   1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

      > **Note**
      >
      > Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

   2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

      Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

      > **Note**
      >
      > Si el proyecto de demostración que implementó contiene datos de entrada de ejecución rellenados previamente, utilice esa entrada para ejecutar la máquina de estado.

   3. Seleccione Iniciar ejecución.
   4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

      Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Código de la máquina de estado de ejemplo

La máquina de estados de este proyecto de ejemplo se integra con AWS Batch Amazon SNS pasando los parámetros directamente a esos recursos.

Explore este ejemplo de máquina de estados para ver cómo Step Functions controla AWS Batch Amazon SNS conectándose al nombre de recurso de Amazon (ARN) en el `Resource` campo y pasándolo `Parameters` a la API del servicio.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

```
{
  "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
    }
  }
}
```

Ejemplo de IAM

Estas políticas de ejemplo AWS Identity and Access Management (IAM) generadas por el proyecto de ejemplo incluyen los privilegios mínimos necesarios para ejecutar la máquina de estados y los recursos relacionados. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM.

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"
        }
    ]
}
```

Para obtener información sobre cómo configurar IAM al utilizar Step Functions con otros AWS servicios, consulte Políticas de IAM para servicios integrados.

Encadenamiento de mensajes de IA con Amazon Bedrock

Este proyecto de ejemplo demuestra cómo puede realizar una integración con Amazon Bedrock para realizar encadenamiento de mensajes de IA. Este proyecto de ejemplo muestra cómo puede crear chatbots de alta calidad utilizando Amazon Bedrock. El proyecto encadena algunos mensajes y los resuelve en la secuencia en que se proporcionan. El encadenamiento de estos mensajes aumenta la capacidad del modelo de lenguaje que se utiliza para ofrecer una respuesta muy precisa.

Este proyecto de ejemplo crea la máquina de estados, los AWS recursos auxiliares y configura los permisos de IAM relacionados. Explore este proyecto de ejemplo para aprender acerca del uso de la integración de servicios optimizada de Amazon Bedrock con máquinas de estado de Step Functions o utilícelo como punto de partida para sus propios proyectos.

Temas

- Paso 2: Ejecutar la máquina de estado

Plantilla de AWS CloudFormation y recursos adicionales

Utilice una plantilla de CloudFormation para implementar este proyecto de muestra. Esta plantilla crea los siguientes recursos en su: Cuenta de AWS

- Una máquina de estado de Step Functions.
- Rol de ejecución para la máquina de estado. Esta función otorga los permisos que su máquina de estado necesita para acceder a otros Servicios de AWS recursos, como la Amazon Bedrock InvokeModelacción.

Requisitos previos

En este proyecto de ejemplo se utiliza el modelo de lenguaje grande (LLM) de Cohere Command. Para ejecutar correctamente este proyecto de ejemplo, debe añadir acceso a este LLM desde la consola de Amazon Bedrock. Para agregar el acceso a modelos, haga lo siguiente:

1. Abra la consola de Amazon Bedrock.
2. En el panel de navegación, elija Acceso a modelos.
3. Elija Administrar el acceso a modelos.
4. Seleccione la casilla situada junto a Cohere.
5. Elegir Solicitar acceso. El Estado de acceso del modelo Cohere se muestra como Acceso concedido.

Paso 1: Crear la máquina de estado y aprovisionar recursos

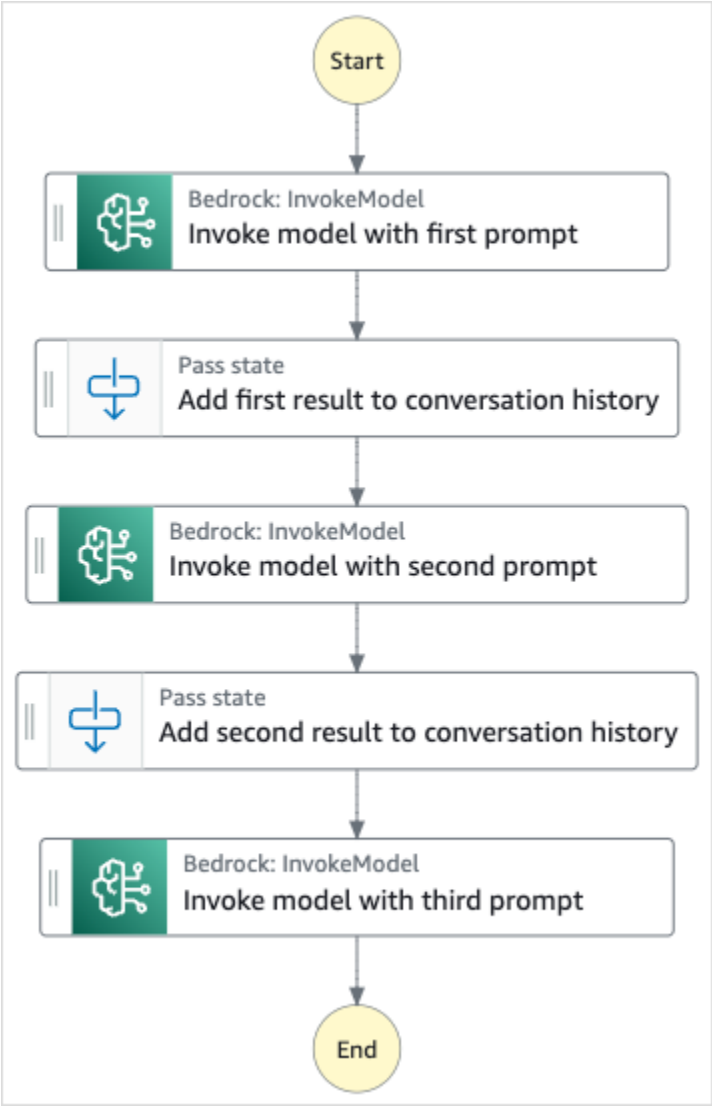
1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. Escriba **`bedrock`** en el cuadro de búsqueda y, a continuación, seleccione Realizar encadenamiento de mensajes de IA con Bedrock en los resultados de búsqueda que aparecen.
3. Elija Siguiente para continuar.
4. Step Functions muestra una lista de las Servicios de AWS utilizadas en el proyecto de muestra que ha seleccionado. También muestra un gráfico del flujo de trabajo para el proyecto de muestra. Implemente este proyecto en su empresa Cuenta de AWS o utilícelo como punto de

partida para crear sus propios proyectos. En función de cómo desee continuar, elija Ejecutar una demostración o Crear a partir de ella.

En este proyecto de muestra se implementan los siguientes recursos:

- ¿Una máquina de AWS Step Functions estados
- Funciones relacionadas AWS Identity and Access Management (IAM)

En la siguiente imagen se ilustra el gráfico del flujo de trabajo para el proyecto de ejemplo Realizar encadenamiento de mensajes de IA con Bedrock:



5. Elija Utilizar plantilla para continuar con la selección.
6. Realice una de las acciones siguientes:

- Si se ha seleccionado Crear a partir de ella, Step Functions crea el prototipo de flujo de trabajo para el proyecto de muestra que ha seleccionado. Step Functions no implementa los recursos que se enumeran en la definición del flujo de trabajo.

  En Modo Diseño de Workflow Studio, arrastre y suelte los estados desde el Navegador de estados para seguir creando su prototipo de flujo de trabajo. Del mismo modo, cambie al Modo Código que proporciona un editor de código integrado similar a VS Code para actualizar la definición (ASL) de Lenguaje de estados de Amazon de su máquina de estado en la consola de Step Functions. Para obtener más información acerca del uso de Workflow Studio para crear máquinas de estados, consulte Usar Workflow Studio.

  > **Important**
  >
  > No olvide actualizar el marcador de posición del nombre de recurso de Amazon (ARN) para los recursos que se utilizan en el proyecto de muestra antes de ejecutar el flujo de trabajo.

- Si seleccionó Ejecutar una demostración, Step Functions crea un proyecto de ejemplo de solo lectura que utiliza una AWS CloudFormation plantilla para implementar los AWS recursos que figuran en esa plantilla en su empresa. Cuenta de AWS

  > **Tip**
  >
  > Seleccione Código para ver la definición de máquina de estados del proyecto de muestra.

  Cuando esté listo, elija Implementar y ejecutar para implementar el proyecto de muestra y crear los recursos.

  El proceso de creación de estos recursos y los permisos de IAM relacionados puede tardar hasta 10 minutos. Mientras se despliegan sus recursos, puede abrir el enlace CloudFormation Stack ID para ver qué recursos se están aprovisionando.

  Una vez que se creen todos los recursos del proyecto de muestra, podrá ver el nuevo proyecto de muestra en la página Máquinas de estado.

> **Important**
>
> Es posible que se apliquen cargos estándar por cada servicio utilizado en la CloudFormation plantilla.

Paso 2: Ejecutar la máquina de estado

1. En la página Máquina de estado, elija su proyecto de muestra.
2. En la página del proyecto de muestra, seleccione Iniciar ejecución.
3. En el cuadro de diálogo Iniciar ejecución, haga lo siguiente:

    1. (Opcional) Para identificar la ejecución, puede especificar un nombre en el cuadro Nombre. De forma predeterminada, Step Functions genera automáticamente un nombre de ejecución único.

        > **Note**
        >
        > Step Functions permite crear nombres para máquinas de estado, ejecuciones, actividades y etiquetas que contengan caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

    2. (Opcional) En el cuadro Entrada, introduzca los valores de entrada en formato JSON para ejecutar el flujo de trabajo.

        Si se ha seleccionado Ejecutar una demostración, no es necesario proporcionar ninguna entrada de ejecución.

    3. Seleccione Iniciar ejecución.
    4. La consola de Step Functions le dirige a una página cuyo título es su ID de ejecución. Esta página se conoce como Detalles de la ejecución. En esta página, puede revisar los resultados de la ejecución a medida que avanza la ejecución o una vez finalizada.

        Para revisar los resultados de la ejecución, elija los estados individuales en la Vista de gráfico y, a continuación, elija las pestañas individuales del panel Detalles del paso para ver los

detalles de cada estado, incluidas la entrada, la salida y la definición, respectivamente. Para obtener más información sobre la ejecución que puede ver en la página Detalles de la ejecución, consulte Página de detalles de ejecución: información general de la interfaz.

Cuotas

AWS Step Functions establece cuotas en los tamaños de determinados parámetros de las máquinas de estado, como la cantidad de acciones de la API durante un período de tiempo determinado o la cantidad de máquinas de estado que puede definir. Aunque estas cuotas se han diseñado para impedir que una máquina de estado configurada erróneamente consuma todos los recursos del sistema, muchas de ellas no son inflexibles.

Para solicitar un aumento de la cuota de servicio, puede elegir una de las siguientes opciones:

- Use la consola de Service Quotas en https://console.aws.amazon.com/servicequotas/. Para obtener información sobre cómo solicitar aumento de cuota mediante la consola Service Quotas, consulte Solicitud de aumento de cuota en la Guía del usuario de Service Quotas.
- Utilice la página Support Center de AWS Management Console para solicitar un aumento de la cuota de los recursos proporcionados por AWS Step Functions región. Para obtener más información, consulte el artículo sobre AWS Service Quotas en la Referencia general de AWS.

> **Note**
>
> Si una determinada etapa de la ejecución de su máquina de estado o actividad tarda demasiado tiempo, puede configurar el tiempo de espera de una máquina de estado para producir un evento de tiempo de espera agotado.

Temas

Cuotas generales

Cuota	Descripción
Nombres en Step Functions	Los nombres de máquinas de estado, ejecuciones y tareas de actividad no deben superar los 80 caracteres. Estos nombres deben ser exclusivos para su cuenta y AWS región, y no deben contener ninguno de los siguientes elementos: • Espacios en blanco • Caracteres comodín (`? *`) • Caracteres entre corchetes (`< > { } [ ]`) • Caracteres especiales (`" # % \ ^ | ~ ` $ & , ; : /`) • Caracteres de control (`\\u0000` - `\\u001f` o `\\u007f` - `\\u009f`). Si la máquina de estado es de tipo rápido, puede proporcionar el mismo nombre a varias ejecuciones de la máquina de estado. Step Functions genera un ARN de ejecución único para cada ejecución de una máquina de estado rápida, incluso aunque varias ejecuciones tengan el mismo nombre. Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividad es, así como etiquetas que contienen caractere s no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimien to de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.

Cuotas relacionadas con las cuentas

Recurso	Cuota predeterminada	Se puede aumentar a
Número máximo de máquinas de estado registradas	10 000	25 000
Número máximo de actividad es registradas	10 000	15.000
Tamaño máximo de solicitud	1 MB por solicitud. Este es el tamaño total de los datos para cada solicitud API de Step Functions, incluido el encabezado de la solicitud y todos los demás datos de solicitud relacionados.	Cuota invariable
Número máximo de ejecucion es abiertas por cuenta	1 000 000 de ejecuciones para cada Cuenta de AWS en cada Región de AWS. Si se excede este valor provocará un error `ExecutionLimitExce eded` . No se aplica a los flujos de trabajo rápidos.	Millones
Número máximo de Map Runs abiertas Una Map Run abierta es una Map Run que se ha iniciado, pero que aún no se ha completado. Las ejecucion es de mapas programad as esperan al momento del MapRunStartedevento hasta que el número total de ejecuciones de mapas	1 000 Esta cuota se aplica al estado Distributed Map.	Cuota invariable

Recurso	Cuota predeterminada	Se puede aumentar a
abiertas sea inferior a la cuota predeterminada de 1000.		
redrives máximas de una Map Run.	1 000 Esta cuota se aplica al estado Distributed Map.	Cuota invariable
Número máximo de ejecuciones secundarias de Map Run en paralelo	10 000	Cuota invariable

Cuotas relacionadas con la tarea HTTP

Las tareas HTTP se limitan mediante un esquema de bucket de token para mantener el ancho de banda del servicio de Step Functions.

Recurso	Tamaño del bucket	Velocidad de reposición por segundo
Tarea HTTP	300	300

En la siguiente tabla se muestra la cuota de duración de una tarea HTTP.

Recurso	Cuota predeterminada
Duración de la tarea HTTP La duración de una tarea HTTP hace referencia al tiempo que tarda una tarea HTTP en enviar una solicitud HTTP y recibir una respuesta.	60 segundos Esta es una cuota rígida que no se puede cambiar.

Cuotas relacionadas con la limitación controlada de estados

Las transiciones de estado de Step Functions se limitan mediante un esquema de bucket de token para mantener el ancho de banda del servicio. Los flujos de trabajo estándar y los flujos de trabajo rápidos tienen una limitación de las transiciones de estado diferente. Las cuotas de flujos de trabajo estándar son cuotas flexibles y se pueden aumentar.

> **Note**
>
> La limitación de la métrica del `StateTransition` servicio se informa como en `ExecutionThrottled` Amazon. CloudWatch Para obtener más información, consulta la `ExecutionThrottled` CloudWatch métrica.

	Standard		Express	
Métricas de servicio	Tamaño del bucket	Velocidad de reposición por segundo	Tamaño del bucket	Velocidad de reposición por segundo
`StateTransition` — Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón) y Europa (Irlanda)	5 000	5 000	Sin límite	Sin límite
`StateTransition` — todas las demás regiones	800	800	Sin límite	Sin límite

Cuotas relacionadas con la limitación controlada de las acciones de la API

Algunas acciones de la API de Step Functions se limitan mediante un esquema de bucket de token para mantener el ancho de banda del servicio. Estas cuotas son cuotas flexibles y se pueden aumentar.

> **Note**
>
> Las cuotas de limitación son por cuenta y por región. AWS
> AWS Step Functions puede aumentar tanto el tamaño de la cubeta como la tasa de recarga en cualquier momento.

	Standard		Express	
Nombre de API	Tamaño del bucket	Velocidad de reposición por segundo	Tamaño del bucket	Velocidad de reposición por segundo
`StartExecution` — Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón) y Europa (Irlanda)	1300	300	6000	6000
`StartExecution` — todas las demás regiones	800	150	6000	6000

Cuota relacionada con la API TestState

Nombre de API	Cuota	Se puede aumentar
TestState	1 transacción por segundo (TPS)	Cuota invariable

Otras cuotas

Estas cuotas son cuotas flexibles y se pueden aumentar.

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nombre de API	Tamaño del bucket	Velocidad de reposición por segundo	Tamaño del bucket	Velocidad de reposición por segundo
`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

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nombre de API	Tamaño del bucket	Velocidad de reposición por segundo	Tamaño del bucket	Velocidad de reposición por segundo
`DescribeStateMachine`	200	20	200	20
`DescribeStateMachineForExecution`	200	1	200	1
`GetActivityTask`	3000	500	1500	300
`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`	3000	500	1500	300
`SendTaskHeartbeat`	3000	500	1500	300

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nombre de API	Tamaño del bucket	Velocidad de reposición por segundo	Tamaño del bucket	Velocidad de reposición por segundo
`SendTaskSuccess`	3000	500	1500	300
StartSync Execution	Las llamadas a la API de ejecución rápidas sincrónica no contribuyen a los límites de capacidad de las cuentas existentes. Step Functions proporciona capacidad bajo demanda y escala automáticamente con una carga de trabajo sostenida. Los picos de carga de trabajo pueden reducirse hasta que haya capacidad disponible. Si se produce una limitación, inténtelo de nuevo después de un tiempo. Para obtener información sobre los flujos de trabajo rápidos sincrónicos, consulte Flujos de trabajo rápidos síncronos y asíncronos.			
`StopExecution`	1 000	200	500	25
`TagResource`	200	1	200	1
`UntagResource`	200	1	200	1
`UpdateStateMachine`	100	1	100	1

Cuotas relacionadas con ejecuciones de máquinas de estado

En la siguiente tabla se describen las cuotas relacionadas con las ejecuciones de máquinas de estado. Las cuotas de ejecución de máquinas de estado son cuotas invariables que no se pueden cambiar, excepto la cuota de Tiempo de retención del historial de ejecuciones.

Cuota	Estándar	Rápido
Tiempo máximo de ejecución	1 año. Si una ejecución dura más de un año como máximo, se producirá un `States.Timeout` error y se emitirá una `ExecutionsTimedOut` CloudWatch métrica.	5 minutos. Si una ejecución dura más de 5 minutos como máximo, se producirá un `States.Timeout` error y se emitirá una `ExecutionsTimedOut` CloudWatch métrica.
Tamaño máximo del historial de ejecución	25 000 eventos en un historial de ejecución de máquinas de un solo estado. Si el historial de ejecuciones alcanza esta cuota, la ejecución non funcionará correctamente. Para evitar esto, consulta Evitar alcanzar la cuota de historial.	Sin límite.
Tiempo máximo de inactividad de ejecución.	1 año (limitado por el tiempo máximo de ejecución).	5 minutos (limitado por el tiempo máximo de ejecución).
Tiempo de retención del historial de ejecución	90 días después del cierre de una ejecución. Transcurrido ese plazo, no podrá ver ni recuperar el historial de ejecuciones. No hay ninguna cuota con respecto al número de ejecuciones cerradas que conserva Step Functions. Para cumplir con los requisitos de conformidad, organizativos o normativos, se puede reducir el periodo de retención del historial de ejecución a	Para ver el historial de ejecuciones, se debe configurar el CloudWatch registro de Amazon Logs. Para obtener más información, consulte Registro mediante CloudWatch Logs.

Cuota	Estándar	Rápido
	30 días mediante el envío de una solicitud de cuota. Para ello, utilice AWS Support Center Console y cree un caso nuevo. El cambio para reducir el período de retención a 30 días es aplicable para cada cuenta de una región.	
Periodo redrivable de ejecución El período Redrivable hace referencia al tiempo durante el cual se puede redrive la ejecución de un determinado flujo de trabajo estándar. Este período comienza el día en que una máquina de estado completa su ejecución.	14 días. Esta cuota rígida se aplica al estado Distributed Map.	Redrive no es compatible actualmente con los flujos de trabajo rápidos.

Cuotas relacionadas con ejecuciones de tarea

En la siguiente tabla se describen las cuotas relacionadas con la ejecución de tareas. Todas estas son cuotas fijas que no se pueden cambiar.

Cuota	Estándar	Rápido
Tiempo máximo de ejecución de la tarea	1 año (limitado por el tiempo máximo de ejecución)	5 minutos (limitado por el tiempo máximo de ejecución)

Cuota	Estándar	Rápido
Tiempo máximo que Step Functions mantiene una tarea en la cola	1 año (limitado por el tiempo máximo de ejecución)	5 minutos (limitado por el tiempo máximo de ejecución)
Número máximo de sondeos de actividades por nombre de recurso de Amazon (ARN)	1000 sondeadores llamando a `GetActivityTask` por cada ARN. Si se supera esta cuota, se produce este error: "The maximum number of workers concurrently polling for activity tasks has been reached." (Se ha alcanzado el número máximo de procesos de trabajo que sondean de forma simultánea las tareas de actividad).	No se aplica a los flujos de trabajo rápidos.
Tamaño máximo de los datos de entrada o salida para una tarea, estado o ejecución	256 KB de datos como cadena codificada en UTF-8. Esa cuota afecta a las tareas (actividad, función de Lambda o servicio integrado), a los datos de salida de los estados o las ejecuciones y a los datos de entrada cuando se programa una tarea, se entra en un estado o se inicia una ejecución.	256 KB de datos como cadena codificada en UTF-8. Esa cuota afecta a las tareas (actividad, función de Lambda o servicio integrado), a los datos de salida de los estados o las ejecuciones y a los datos de entrada cuando se programa una tarea, se entra en un estado o se inicia una ejecución.

Cuotas relacionadas con versiones y alias

Recurso	Cuota predeterminada
Número máximo de versiones de máquinas de estado publicadas	1000 para cada máquina de estado. Para solicitar un aumento de este límite flexible, utilice la página Centro de soporte de AWS Management Console.
Número máximo de alias de máquinas de estado	100 para cada máquina de estado. Para solicitar un aumento de este límite flexible, utilice la página Centro de soporte de AWS Management Console.

Restricciones relacionadas con el etiquetado

Tenga en cuenta estas restricciones cuando etiquete los recursos de Step Functions.

> **Note**
>
> Las restricciones de etiquetado no se pueden aumentar como otras cuotas.

Restricción	Descripción
Número máximo de etiquetas por recurso	50
Longitud máxima de clave	128 caracteres Unicode en UTF-8
Longitud máxima de valor	256 caracteres Unicode en UTF-8
Restricción de prefijo	No utilice el `aws:` prefijo en los nombres o valores de las etiquetas porque está reservado para su AWS uso. Los nombres y valores de etiquetas que tienen este prefijo no se

Restricción	Descripción
	pueden editar ni eliminar. Las etiquetas que tengan este prefijo no cuentan para la cuota de etiquetas por recurso.
Restricciones de caracteres	Las etiquetas solo pueden contener letras Unicode, dígitos, espacios en blanco o estos símbolos: `_ . : / = + - @`.

Inicio de sesión y supervisión AWS Step Functions

El registro y la supervisión son importantes para mantener la fiabilidad, la disponibilidad y el rendimiento de Step Functions y sus AWS soluciones. Hay varias herramientas disponibles para usar con Step Functions:

> **Tip**
>
> Para implementar un ejemplo de flujo de trabajo Cuenta de AWS y aprender a monitorear las métricas, los registros y los rastros de la ejecución del flujo de trabajo, consulte el Módulo 12: Observabilidad de The AWS Step Functions Workshop.

Temas

Supervisión de las funciones de Step Functions mediante CloudWatch

La supervisión es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de AWS Step Functions sus AWS soluciones. Debe recopilar la mayor cantidad de datos de supervisión de los AWS servicios que utilice para poder depurar errores multipuntuales. Antes de empezar la monitorización de Step Functions, debe crear un plan de monitorización que responda a las siguientes preguntas:

- ¿Cuáles son los objetivos de la supervisión?
- ¿Qué recursos va a supervisar?
- ¿Con qué frecuencia va a supervisar estos recursos?

- ¿Qué herramientas de monitoreo va a utilizar?
- ¿Quién se encargará de realizar las tareas de monitoreo?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso consiste en establecer un punto de referencia para el rendimiento normal de en su entorno. Para ello, mida el rendimiento en diversas ocasiones y con diferentes condiciones de carga. Cuando monitoree Step Functions, debe tener en cuenta el almacenamiento de los datos de monitoreo históricos. Estos datos pueden servirle como referencia para compararlos con los datos de rendimiento actuales, para identificar los patrones de rendimiento normales y las anomalías del rendimiento, y para idear métodos de solución de problemas.

Por ejemplo, con Step Functions, puedes controlar cuántas actividades o AWS Lambda tareas fallan debido al tiempo de espera de un latido. Si el rendimiento se sale de la referencia establecida, es posible que tenga que cambiar el intervalo de latidos.

Para establecer un punto de referencia debe, como mínimo, monitorizar las siguientes métricas:

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

En las siguientes secciones se describen las métricas que Step Functions proporciona a Amazon CloudWatch. Puede utilizarlas para supervisar las máquinas de estado y las actividades, así como para establecer alarmas en los valores que actúan como umbral. Puede ver las métricas mediante AWS Management Console.

Métricas que informan sobre un intervalo de tiempo

Algunas de las CloudWatch métricas de Step Functions son intervalos de tiempo, siempre medidos en milisegundos. Estas métricas, por lo general, se corresponden con fases de ejecución en las que puede definir tiempos de espera para las máquinas de estado, las actividades y las funciones de Lambda utilizando nombres descriptivos.

Por ejemplo, la métrica `ActivityRunTime` mide el tiempo que tarda una actividad en completarse una vez que comienza su ejecución Puede establecer un tiempo de espera con este mismo período de tiempo.

En la CloudWatch consola, puede obtener los mejores resultados si elige el promedio como estadística de visualización para las métricas de intervalos de tiempo.

Métricas que informan de un recuento

Algunas de las CloudWatch métricas de Step Functions muestran los resultados como un recuento. Por ejemplo, `ExecutionsFailed` registra el número o recuento de ejecuciones con errores de la máquina de estado.

Step Functions emite dos `ExecutionsStarted` métricas por cada ejecución de una máquina de estado. Esto hace que la SampleCountestadística de la `ExecutionsStarted` métrica muestre el valor de 2 para cada ejecución de una máquina de estados. La SampleCount estadística muestra `ExecutionStarted=1` y `ExecutionStarted=0` cuándo se completa la ejecución.

> **Tip**
>
> Recomendamos seleccionar Suma como estadística de visualización para las métricas que indican un recuento en la CloudWatch consola.

Métricas de ejecución

El espacio de `AWS/States` nombres incluye las siguientes métricas para todas las ejecuciones de Step Functions. Se trata de métricas adimensionales que se aplican a toda tu cuenta en una región.

Métrica	Descripción
`OpenExecutionCount`	Número aproximado de ejecuciones actualmente abiertas: flujos de trabajo que se encuentran actualmente en curso en su cuenta. El objetivo es proporcionar información sobre cuándo tus flujos de trabajo se acercan al límite máximo de ejecución, a fin de evitar ExecutionLimitExceedederrores al realizar llamadas

Métrica	Descripción
	`StartExecution` o en el caso de los flujos `RedriveExecution` de trabajo estándar. La métrica depende de las transiciones de estado del flujo de trabajo activo, por lo que, en niveles bajos, es posible que la estimación no se ajuste al recuento observado del flujo de trabajo en ejecución.
`OpenExecutionLimit`	Número máximo de ejecuciones abiertas. Para obtener más información, consulte Cuotas relacionadas con las cuentas. Este límite no se aplica a los flujos de trabajo de Express.

Métricas de ejecución para máquinas de estados con versión o alias

Cuando ejecutas una ejecución de máquina de estados con una versión o un alias, Step Functions emite las siguientes métricas. La `ExecutionThrottled` métrica solo se emitirá en caso de ejecución limitada. Estas métricas incluirán una `StateMachineArn` para identificar una máquina de estados específica.

Métrica	Descripción
`ExecutionTime`	Intervalo, en milisegundos, entre el momento en que se inicia la ejecución y el momento en que se cierra.
`ExecutionThrottled`	Número de `StateEntered` eventos y reintentos que se han limitado. Está relacionado con la limitación controlada de `StateTransition` . Para obtener más información, consulte Cuotas relacionadas con la limitación controlada de estados.
`ExecutionsAborted`	Número de ejecuciones abortadas o canceladas.
`ExecutionsFailed`	Número de ejecuciones fallidas.
`ExecutionsStarted`	Número de ejecuciones iniciadas.
`ExecutionsSucceeded`	Número de ejecuciones completadas satisfactoriamente.

Métrica	Descripción
`ExecutionsTimedOut`	Número de ejecuciones cuyo tiempo de espera se ha agotado por cualquier motivo.

Métricas de ejecución para flujos de trabajo rápidos

El espacio de nombres `AWS/States` incluye las siguientes métricas para las ejecuciones de flujos de trabajo rápidos de Step Functions.

Métrica	Descripción
`ExpressExecutionMemory`	La memoria total consumida por un flujo de trabajo rápido.
`ExpressExecutionBilledDuration`	La duración por la que se cobra un flujo de trabajo rápido.
`ExpressExecutionBilledMemory`	La cantidad de memoria consumida por la que se cobra un flujo de trabajo rápido.

Redrive de métricas de ejecución para flujos de trabajo estándar

Cuando redrive la ejecución de una máquina de estado, Step Functions emite las siguientes métricas.

La métrica `Executions*` se emite para todas las ejecuciones redriven. Por ejemplo, supongamos que una ejecución redriven se anula. Esta ejecución emitirá puntos de datos distintos de cero para `RedrivenExecutionsAborted` y `ExecutionsAborted`.

Métrica	Descripción
`ExecutionsRedriven`	Número de redriven ejecuciones.
`RedrivenExecutionsAborted`	Número de redriven ejecuciones que se cancelan o finalizan.

Métrica	Descripción
`RedrivenExecutionsTimedOut`	Número de redriven ejecuciones cuyo tiempo de espera se ha agotado por cualquier motivo.
`RedrivenExecutionsSucceeded`	Número de redriven ejecuciones que se completaron satisfact oriamente.
`RedrivenExecutionsFailed`	Número de redriven ejecuciones fallidas.

Dimensión para métricas de ejecución de Step Functions

Dimensión	Descripción
`StateMachineArn`	El nombre de recurso de Amazon (ARN) de la máquina de estado de la ejecución en cuestión.

Dimensiones de las ejecuciones con una versión

Dimensión	Descripción
`StateMachineArn`	El nombre de recurso de Amazon (ARN) de la máquina de estado cuya ejecución inició una versión.
`Version`	Versión de la máquina de estado que se utiliza para iniciar la ejecución.

Dimensiones de las ejecuciones con un alias

Dimensión	Descripción
`StateMachineArn`	El nombre de recurso de Amazon (ARN) de la máquina de estado cuya ejecución inició un alias.

Dimensión	Descripción
`Alias`	Alias de máquina de estado que se utiliza para iniciar la ejecución.

Métricas de recuento de recursos para versiones y alias

El espacio de nombres `AWS/States` incluye las siguientes métricas para el recuento de versiones y alias de una máquina de estado.

Métrica	Descripción
AliasCount	Número de alias creados para la máquina de estados. Puede crear hasta 100 alias para cada máquina de estado.
VersionCount	Número de versiones publicadas para la máquina de estados. Puede publicar hasta 1000 versiones de una máquina de estado.

Dimensión para métricas de recuento de recursos para versiones y alias

Dimensión	Descripción
`ResourceArn`	El nombre de recurso de Amazon (ARN) de la máquina de estado con un alias o una versión.

Métricas de actividad

El espacio de nombres `AWS/States` incluye las siguientes métricas para las actividades de Step Functions.

Métrica	Descripción
`ActivityRunTime`	Intervalo, en milisegundos, entre el momento en que se inicia la actividad y el momento en que se cierra.
`ActivityScheduleTime`	Intervalo, en milisegundos, durante el que la actividad permanece en el estado programado.
`ActivityTime`	Intervalo, en milisegundos, entre el momento en que se programa la actividad y el momento en que se cierra.
`ActivitiesFailed`	Número de actividades fallidas.
`ActivitiesHeartbeatTimedOut`	Número de actividades cuyo tiempo de espera se interrumpe debido al tiempo de espera de un latido.
`ActivitiesScheduled`	Número de actividades programadas.
`ActivitiesStarted`	Número de actividades iniciadas.
`ActivitiesSucceeded`	Número de actividades completadas satisfactoriamente.
`ActivitiesTimedOut`	Número de actividades cuyo tiempo de espera se agota al finalizar.

Dimensión para métricas de actividad de Step Functions

Dimensión	Descripción
`ActivityArn`	El ARN de la actividad.

Métricas de función de Lambda

El espacio de nombres `AWS/States` incluye las siguientes métricas para las funciones de Lambda de Step Functions.

Métrica	Descripción
`LambdaFunctionRunTime`	Intervalo, en milisegundos, entre el momento en que se inicia la función Lambda y el momento en que se cierra.
`LambdaFunctionScheduleTime`	Intervalo, en milisegundos, durante el que la función Lambda permanece en el estado programado.
`LambdaFunctionTime`	Intervalo, en milisegundos, entre el momento en que se programa la función Lambda y el momento en que se cierra.
`LambdaFunctionsFailed`	Número de funciones Lambda fallidas.
`LambdaFunctionsScheduled`	Número de funciones Lambda programadas.
`LambdaFunctionsStarted`	Número de funciones Lambda iniciadas.
`LambdaFunctionsSucceeded`	Número de funciones Lambda completadas correctamente.
`LambdaFunctionsTimedOut`	Número de funciones Lambda cuyo tiempo de espera se agota al cerrarse.

Dimensión para métricas de la función de Lambda de Step Functions

Dimensión	Descripción
`LambdaFunctionArn`	El ARN de la función de Lambda.

> **Note**
>
> Las métricas de la función de Lambda se emiten para los estados Task que especifican el ARN de la función de Lambda en el campo `Resource`. Los estados Task que utilizan `"Resource": "arn:aws:states:::lambda:invoke"` emiten Métricas de integración

de servicios en su lugar. Para obtener más información, consulte Invocar Lambda con Step Functions.

Métricas de integración de servicios

El espacio de nombres `AWS/States` incluye las siguientes métricas para las integraciones de servicio de Step Functions. Para obtener más información, consulte Uso AWS Step Functions con otros servicios.

Métrica	Descripción
`ServiceIntegrationRunTime`	Intervalo, en milisegundos, entre el momento en que se inicia la tarea de servicio y el momento en que se cierra.
`ServiceIntegrationScheduleTime`	Intervalo, en milisegundos, durante el que la tarea de servicio permanece en el estado programado.
`ServiceIntegrationTime`	Intervalo, en milisegundos, entre el momento en que se programa la tarea de servicio y el momento en que se cierra.
`ServiceIntegrationsFailed`	Número de tareas de servicio fallidas.
`ServiceIntegrationsScheduled`	Número de tareas de servicio programadas.
`ServiceIntegrationsStarted`	Número de tareas de servicio iniciadas.
`ServiceIntegrationsSucceeded`	Número de tareas de servicio completadas correctamente.
`ServiceIntegrationsTimedOut`	Número de tareas de servicio cuyo tiempo de espera se agota al cerrarse.

Dimensión para métricas de integración de servicios de Step Functions

Dimensión	Descripción
`ServiceIntegration ResourceArn`	EL ARN de recurso del servicio integrado.

Métricas de servicios

El espacio de nombres `AWS/States` incluye las siguientes métricas para el servicio de Step Functions.

Métrica	Descripción
`ThrottledEvents`	Recuento de solicitudes que se han limitado.
`ProvisionedBucketS ize`	Recuento de solicitudes disponibles por segundo.
`ProvisionedRefillR ate`	Recuento de solicitudes por segundo que se permiten en el depósito.
`ConsumedCapacity`	Recuento de solicitudes por segundo.

Dimensión para métricas de servicios de Step Functions

Dimensión	Descripción
`ServiceMetric`	Filtra datos para mostrar las métricas de transiciones de estado.

Métricas de API

El espacio de nombres `AWS/States` incluye las siguientes métricas para la API de Step Functions.

Métrica	Descripción
`ThrottledEvents`	Recuento de solicitudes que se han limitado.
`ProvisionedBucketSize`	Recuento de solicitudes disponibles por segundo.
`ProvisionedRefillRate`	Recuento de solicitudes por segundo que se permiten en el depósito.
`ConsumedCapacity`	Recuento de solicitudes por segundo.

Dimensión para métricas de la API de Step Functions

Dimensión	Descripción
`APIName`	Filtra datos a una API con el nombre de API especificado.

Entrega de CloudWatch métricas con el mejor esfuerzo

Las métricas de CloudWatch se entregan según el "mejor esfuerzo", es decir, en la medida que sea posible.

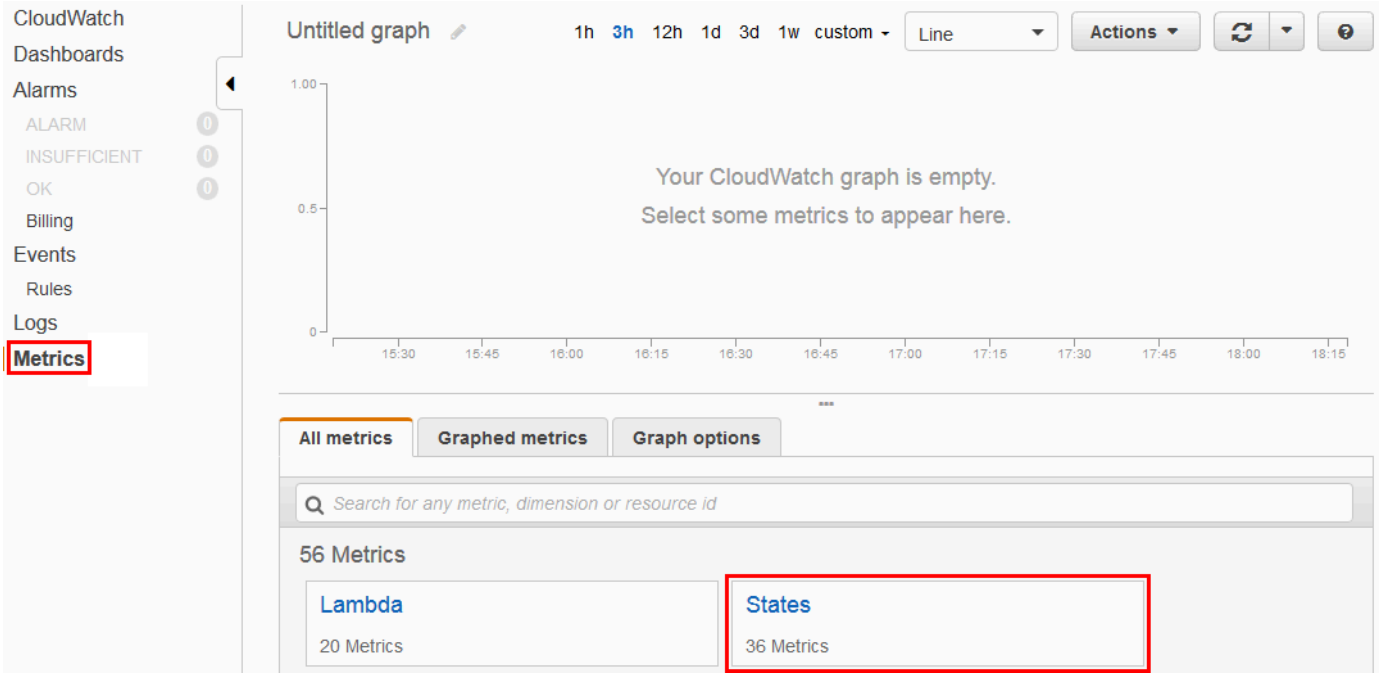
La integridad y la puntualidad de las métricas no están garantizadas. Es posible que el punto de datos de una solicitud determinada se envíe con una marca temporal posterior al momento en el que la solicitud se ha procesado realmente. Es posible que el punto de datos se demore un minuto antes de estar CloudWatch disponible o que no se entregue en absoluto. CloudWatchLas métricas de solicitudes le dan una idea de las ejecuciones de las máquinas de estado prácticamente en tiempo real. No pretende ser un recuento completo de todas las métricas relacionadas con la ejecución.

Dado que esta característica funciona en la medida de lo posible, los informes disponibles en el panel de Administración de facturación y costos podrían incluir una o varias solicitudes de acceso que no aparecen en las métricas de la ejecución.

Visualización de métricas para Step Functions

1. Inicie sesión en la CloudWatch consola AWS Management Console y ábrala.

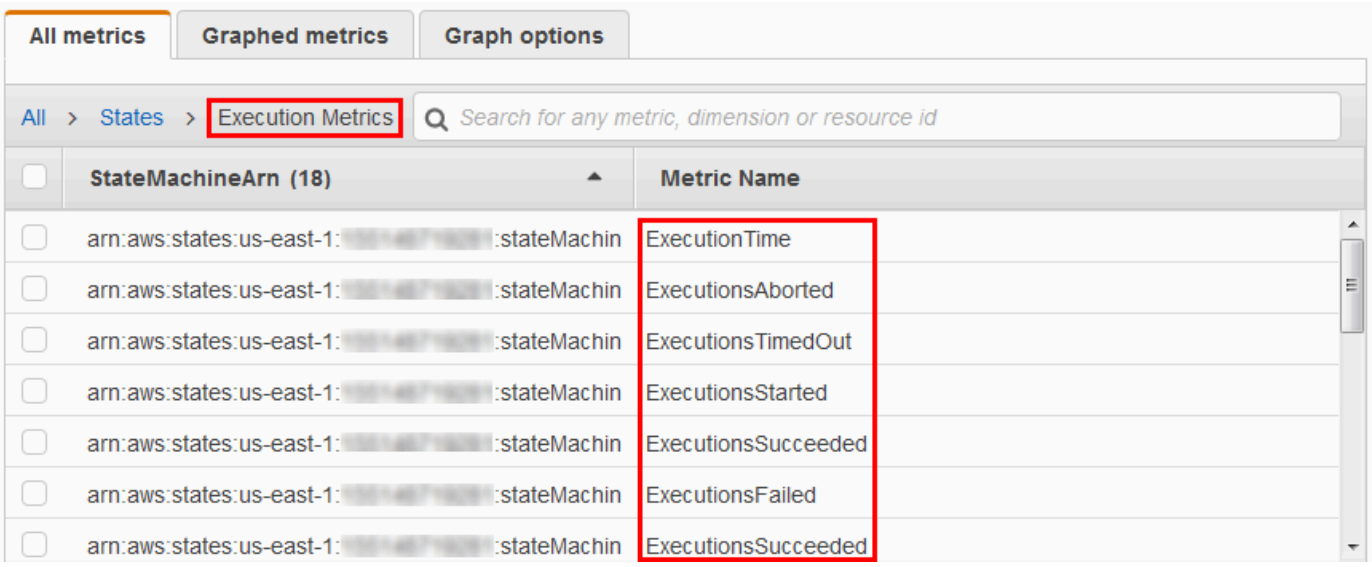
2. Elija Métricas) y, en la pestaña Todas las métricas, elija Estados.



Si recientemente realizó alguna ejecución, podrá ver hasta cuatro tipos de métricas:

- Métricas de ejecución
- Métricas de funciones de actividades
- Métricas de función de Lambda
- Métricas de integración de servicios

3. Elija un tipo de métrica para ver una lista de métricas.

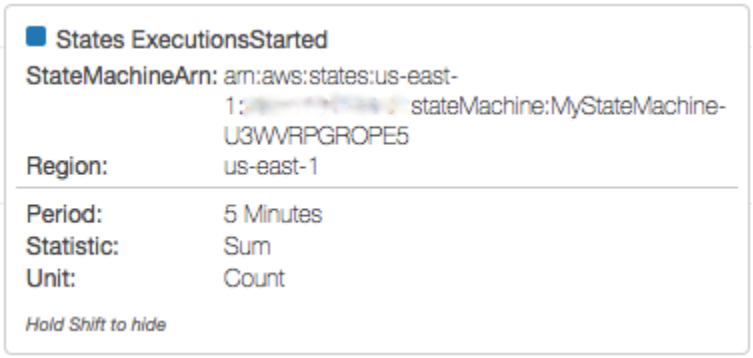


- Para ordenar tus métricas por nombre de métrica o usa StateMachineArnlos encabezados de las columnas.
- Para ver los gráficos de una métrica, active la casilla que aparece en la lista junto a la métrica. Puede cambiar los parámetros del gráfico utilizando los controles de intervalo de tiempo situados encima del gráfico.

  Puede elegir intervalos de tiempo personalizados que utilicen valores relativos o absolutos (fechas y horas específicas). También puede utilizar la lista desplegable para mostrar los valores como líneas, áreas apiladas o números (valores).
- Para ver los detalles de un gráfico, coloque el cursor sobre el código de color de la métrica que aparece debajo del gráfico.

  Se mostrarán los detalles de la métrica.

Para obtener más información sobre cómo trabajar con CloudWatch métricas, consulta Uso de Amazon CloudWatch Metrics en la Guía del CloudWatch usuario de Amazon.

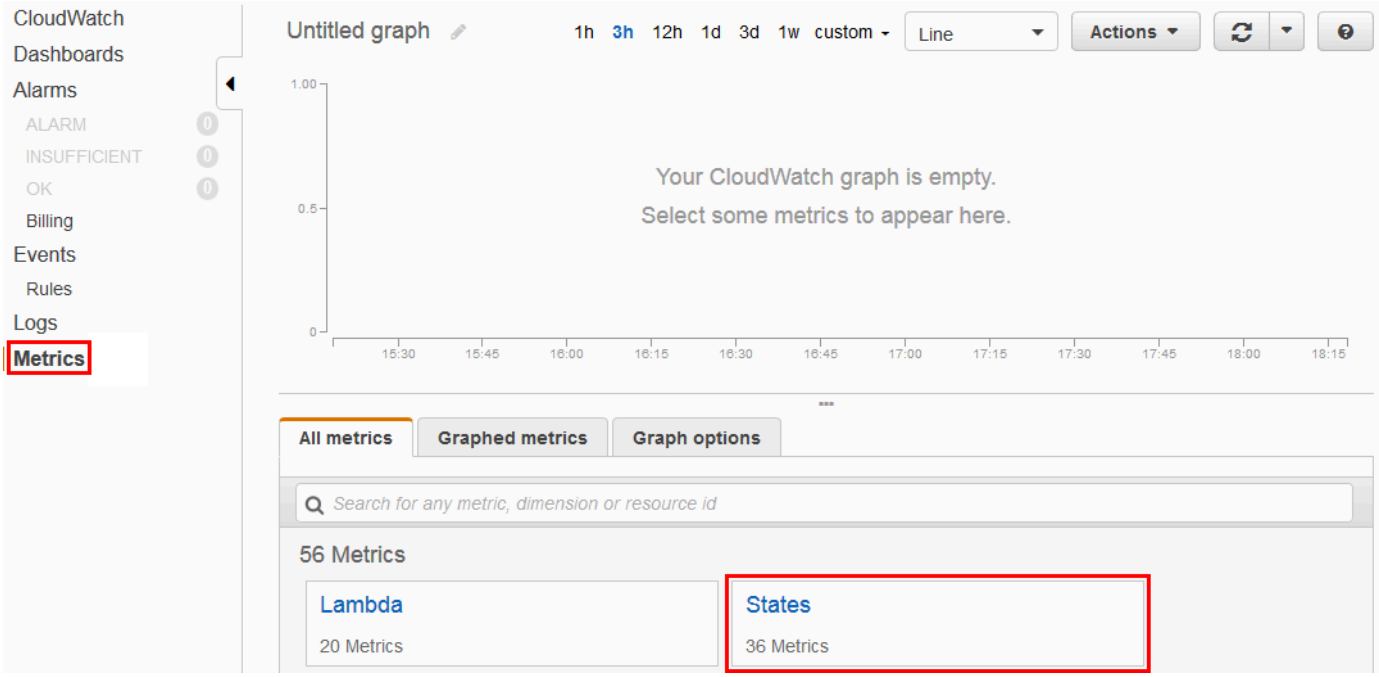
Configuración de alarmas para Step Functions

Puedes usar CloudWatch las alarmas de Amazon para realizar acciones. Por ejemplo, si desea saber cuándo se alcanza el umbral de una alarma, puede configurar una alarma que envíe una notificación a un tema de Amazon SNS o un correo electrónico cuando la métrica `StateMachinesFailed` supere un determinado umbral.

Para configurar una alarma en una métrica

1. Inicia sesión en la CloudWatch consola AWS Management Console y ábrela.

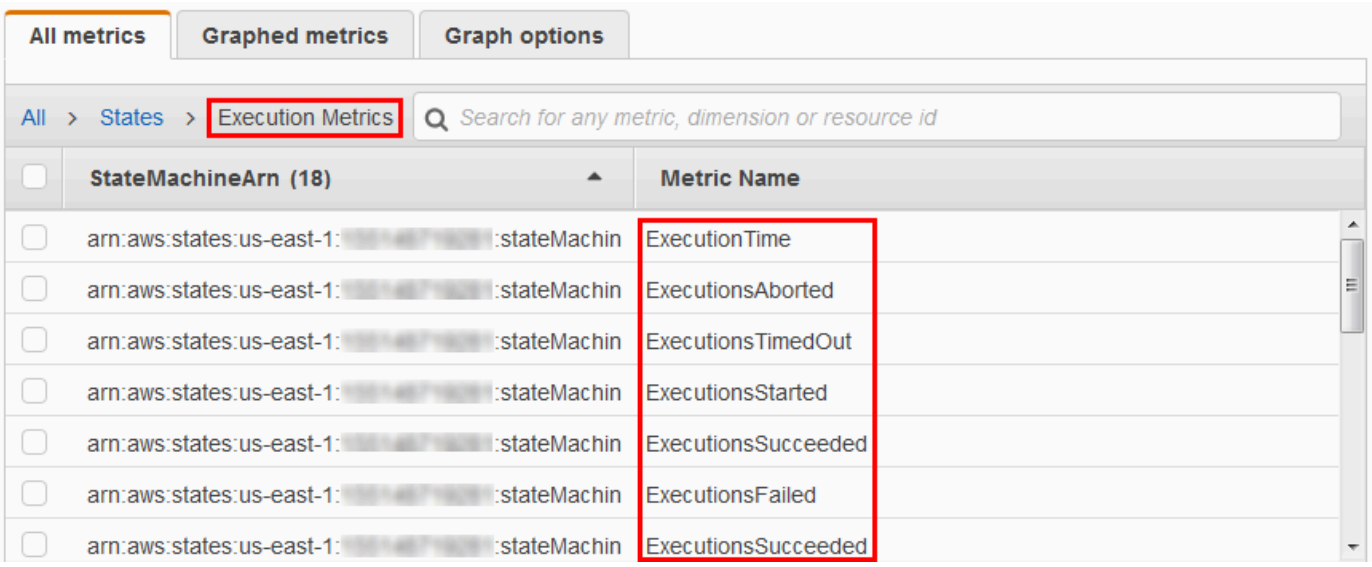
2. Elija Métricas) y, en la pestaña Todas las métricas, elija Estados.



Si recientemente realizó alguna ejecución, podrá ver hasta cuatro tipos de métricas:

- Métricas de ejecución
- Métricas de funciones de actividades
- Métricas de función de Lambda
- Métricas de integración de servicios

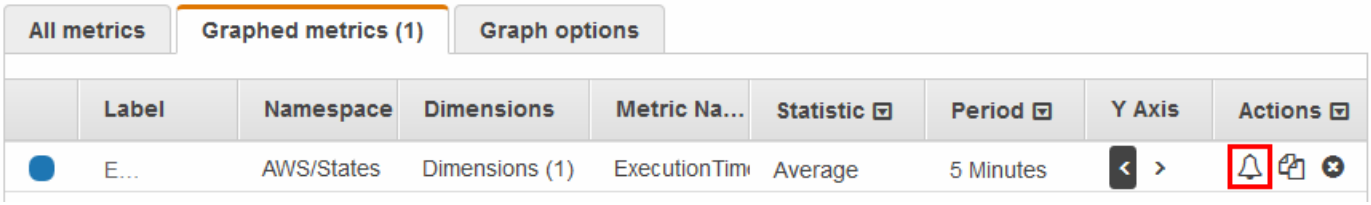
3. Elija un tipo de métrica para ver una lista de métricas.



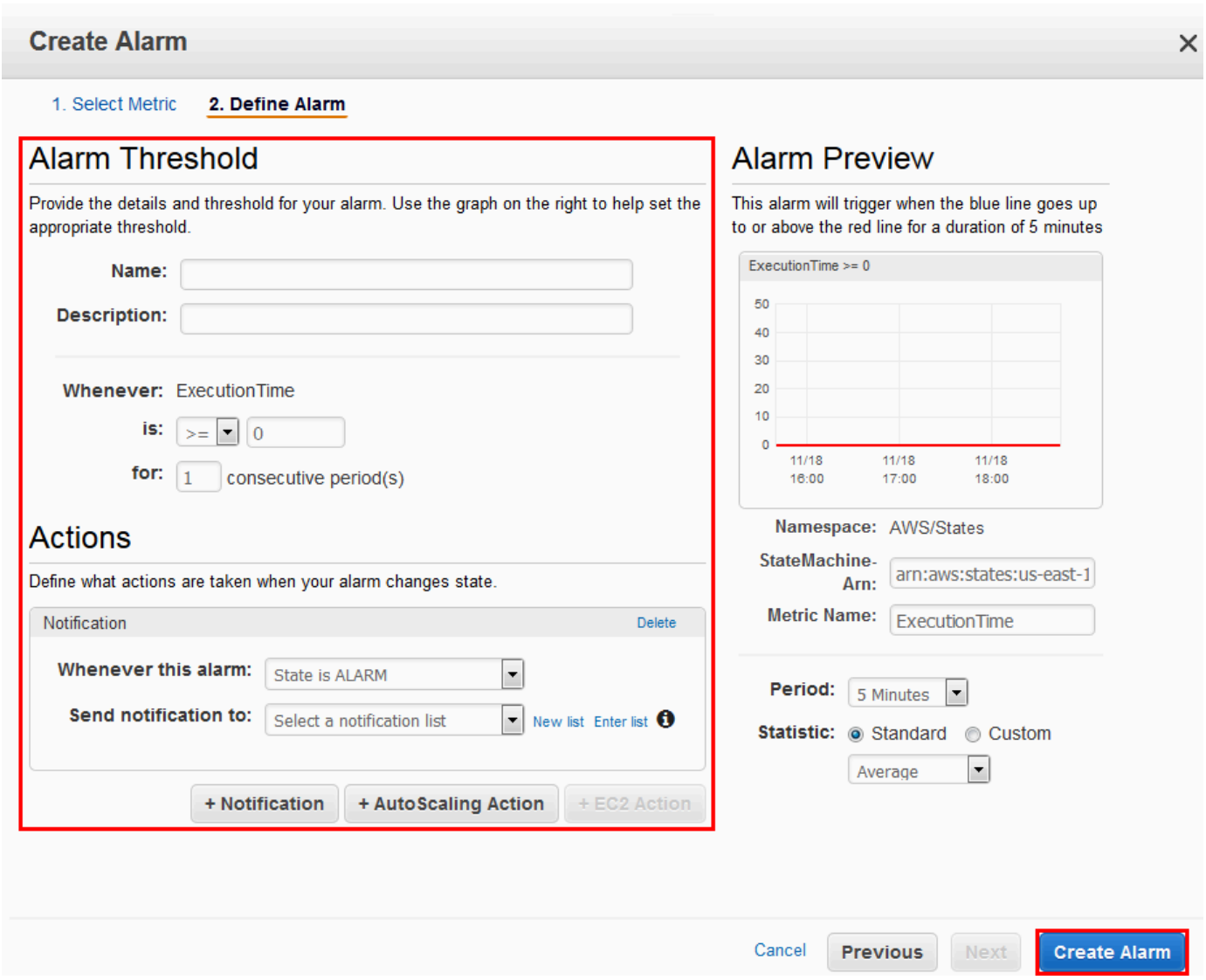
4. Elija una métrica y, a continuación, elija Métricas diagramadas.
5. Seleccione el icono

   que aparece junto a una de las métricas de la lista.



Se muestra la página Crear alarma.



6. Especifique los valores de Umbral de alarma y Acciones y, a continuación, elija Crear alarma.

Para obtener más información sobre la configuración y el uso de CloudWatch alarmas, consulte Creación de CloudWatch alarmas de Amazon en la Guía del CloudWatch usuario de Amazon.

EventBridge (CloudWatch Eventos) para cambios en el estado de ejecución de Step Functions

Amazon EventBridge es un AWS servicio que te permite responder a los cambios de estado de un AWS recurso. Por ejemplo, puede responder a los cambios en el estado de ejecución de un flujo de trabajo estándar de Step Functions de las dos maneras siguientes: EventBridge

- Puede configurar EventBridge reglas para que reaccionen a los eventos que se emiten cuando cambia el estado de ejecución de una máquina de estados de Step Functions. Esto le permite monitorizar los flujos de trabajo sin tener que sondear constantemente mediante la API DescribeExecution. En función de los cambios en las ejecuciones de máquinas de estado, puede utilizar un EventBridge objetivo para iniciar nuevas ejecuciones de máquinas de estado, llamar a AWS Lambda funciones, publicar mensajes en temas del Amazon Simple Notification Service (Amazon SNS) y mucho más.
- También puede configurar una máquina de estados Step Functions como destino en EventBridge. Esto permite activar la ejecución de un flujo de trabajo de Step Functions en respuesta a un evento de otro servicio de AWS .

  Para obtener más información, consulta la Guía del EventBridge usuario de Amazon.

Sin embargo, Express Workflows no emite eventos a EventBridge. Para supervisar la ejecución de un flujo de trabajo exprés, puede utilizar CloudWatch los registros. Para ello, en la página detalles de la ejecución de la máquina de estado, elija las pestañas Monitorización y Registro. En la pestaña Supervisión, puede ver CloudWatch las métricas de los eventos, como la duración de la ejecución, los errores de ejecución y la memoria facturada. En la pestaña Registro, puede ver los registros recientes y la configuración de registro.

> **Tip**
>
> Para utilizar un ejemplo de un flujo de trabajo exprés Cuenta de AWS y aprender a monitorizar los flujos de trabajo exprés, consulte el módulo Supervisión de los flujos de trabajo exprés de The AWS Step Functions Workshop.

EventBridge cargas útiles

Un EventBridge evento puede contener una propiedad de entrada en su definición. En el caso de algunos eventos, un EventBridge evento también puede contener una propiedad de salida en su definición.

- Si la entrada de escape y la salida de escape combinadas enviadas EventBridge superan los 248 KB, se excluirá la entrada. Del mismo modo, si la salida especificada como secuencia de escape supera los 248 KB, la salida se excluye. Esto es el resultado de las cuotas de EventBridge eventos.
- Puede determinar si una carga útil se ha truncado con las propiedades `inputDetails` y `outputDetails`. Para obtener más información, consulte el tipo de datos `CloudWatchEventsExecutionDataDetails`.
- En el caso de los flujos de trabajo estándar, puede ver las entradas y salidas completas utilizando DescribeExecution.
- `DescribeExecution` no está disponible para flujos de trabajo rápidos. Si desea ver la entrada/salida completa, puede encapsular el flujo de trabajo rápido en un flujo de trabajo estándar. Otra opción consiste en utilizar ARN de Amazon S3. Para obtener información acerca del uso de ARN, consulte the section called “Utilizar los ARN de Amazon S3 en lugar de pasar cargas de gran tamaño”.

Temas

- Ejemplos de eventos de Step Functions
- Enrutar un evento de Step Functions a EventBridge la EventBridge consola

Ejemplos de eventos de Step Functions

Los siguientes son ejemplos de cómo Step Functions envía eventos a EventBridge:

Temas

- Ejecución iniciada
- Ejecución correcta
- Ejecución errónea
- Execution con tiempo de espera agotado
- Ejecución anulada

En todos los casos, la sección `detail` de los datos del evento proporciona la misma información que la API `DescribeExecution`. El campo `status` indica el estado de la ejecución en el momento en el que se envió el evento, que será `RUNNING`, `SUCCEEDED`, `FAILED`, `TIMED_OUT` o `ABORTED` en función del evento emitido.

Ejecución iniciada

```
{
    "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
    }
}
```

Ejecución correcta

```
{
    "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
        }
    }
}
```

Ejecución errónea

```
{
    "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
    }
}
```

Execution con tiempo de espera agotado

```
{
    "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
    }
}
```

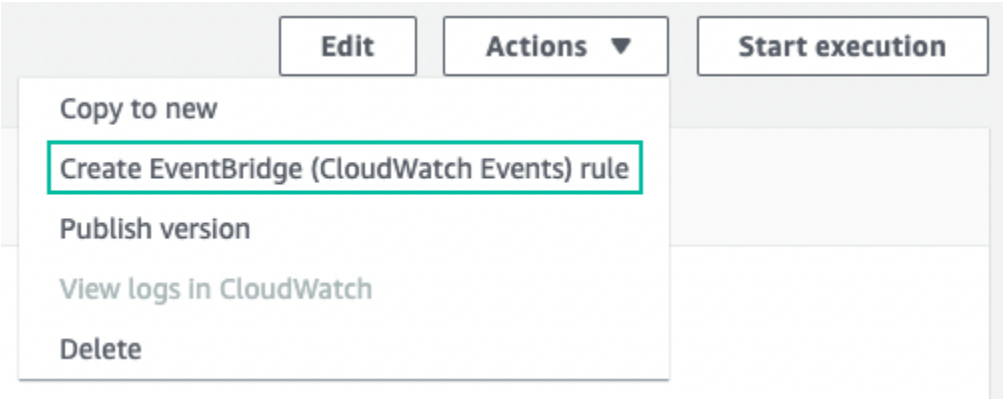
Ejecución anulada

```
{
    "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
    }
}
```

Enrutar un evento de Step Functions a EventBridge la EventBridge consola

Siga las instrucciones siguientes para aprender a activar la ejecución de una máquina de estado de Step Functions cada vez que una máquina de estado de Step Functions específica termine de ejecutarse correctamente. Utiliza la EventBridge consola de Amazon para especificar la máquina de estados cuya ejecución desea activar.

1. En la página de detalles de una máquina de estados, elija Acciones y, a continuación, elija Crear regla EventBridge (CloudWatch eventos).



   También puede abrir la EventBridge consola en https://console.aws.amazon.com/events/. En el panel de navegación, en Buses, elija Reglas.
2. Elija Crear regla. Esto abre la página Definir detalle de la regla.
3. Introduzca un Nombre para la regla (por ejemplo, `StepFunctionsEventRule`) y, si lo desea, introduzca una Descripción para la regla.
4. Para Bus de eventos y Tipo de regla, mantenga las selecciones predeterminadas.
5. Elija Siguiente. Se abrirá la página Crear un patrón de eventos.
6. En Fuente del evento, mantenga la selección predeterminada de AWS eventos o eventos EventBridge asociados.
7. Mantenga las selecciones predeterminadas para las secciones Ejemplo de evento y Método de creación.
8. En Patrón de evento, haga lo siguiente:

   a. En la lista desplegable Origen del evento, mantenga la selección predeterminada de servicios de AWS .

   b. En la lista desplegable Servicio de AWS , seleccione Step Functions.

c. En la lista desplegable Tipo de evento, seleccione Cambio de estado de ejecución de Step Functions.

d. (Opcional) Configure un estado específico, un Nombre de recurso de Amazon (ARN) de la máquina de estado o un ARN de ejecución. Para este procedimiento, elija Estado(s) específico(s) y, a continuación, elija SUCCEEDED en la lista desplegable.

9. Seleccione Siguiente. Se abrirá la página Seleccionar destinos.
10. En Target types, mantenga la selección predeterminada de servicio de AWS .
11. En la lista desplegable Seleccione un objetivo, elija un AWS servicio. Por ejemplo, puede lanzar una función de Lambda o ejecutar una máquina de estado de Step Functions. Para este procedimiento, elija Máquina de estado de Step Functions.
12. En la lista desplegable Máquina de estados, elija una máquina de estado.
13. En Rol de ejecución, mantenga la selección predeterminada de Crear nuevo rol para este recurso específico.
14. Seleccione Siguiente. Se abrirá la página Configurar etiquetas.
15. vuelva a seleccionar Siguiente. Se abrirá la página Revisar y crear.
16. Revise los detalles de la regla y elija Crear regla.

    Se crea la regla y se muestra la página Reglas, en la que se muestran todas tus EventBridge reglas de Amazon.

Grabación de llamadas a la API con AWS CloudTrail

AWS Step Functions está integrado con AWS CloudTrailun servicio que proporciona un registro de las acciones realizadas por un usuario, rol o un Servicio de AWS. CloudTrail captura todas las llamadas a la API Step Functions como eventos. Las llamadas capturadas incluyen llamadas desde la Step Functions consola y llamadas en código a las operaciones de la Step Functions API. Con la información recopilada por CloudTrail, puede determinar a qué solicitud se realizó Step Functions, la dirección IP desde la que se realizó la solicitud, cuándo se realizó y detalles adicionales.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales del usuario raíz o del usuario.
- Si la solicitud se realizó en nombre de un usuario de IAM Identity Center.

- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro Servicio de AWS.

CloudTrail está activa en tu cuenta Cuenta de AWS al crear la cuenta y automáticamente tienes acceso al historial de CloudTrail eventos. El historial de CloudTrail eventos proporciona un registro visible, consultable, descargable e inmutable de los últimos 90 días de eventos de gestión registrados en un. Región de AWSPara obtener más información, consulte Uso del historial de CloudTrail eventos en la Guía del usuario.AWS CloudTrail La visualización del historial de eventos no conlleva ningún CloudTrail cargo.

Para tener un registro continuo de los eventos de Cuenta de AWS los últimos 90 días, crea un almacén de datos de eventos de senderos o CloudTraillagos.

CloudTrail senderos

Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. Todos los senderos creados con él AWS Management Console son multirregionales. Puede crear un registro de seguimiento de una sola región o de varias regiones mediante la AWS CLI. Se recomienda crear un sendero multirregional, ya que puedes capturar toda la actividad de tu Regiones de AWS cuenta. Si crea un registro de seguimiento de una sola región, solo podrá ver los eventos registrados en la Región de AWS del registro de seguimiento. Para obtener más información acerca de los registros de seguimiento, consulte Creación de un registro de seguimiento para su Cuenta de AWS y Creación de un registro de seguimiento para una organización en la Guía del usuario de AWS CloudTrail .

Puede enviar una copia de sus eventos de administración en curso a su bucket de Amazon S3 sin coste alguno CloudTrail mediante la creación de una ruta; sin embargo, hay cargos por almacenamiento en Amazon S3. Para obtener más información sobre CloudTrail los precios, consulte AWS CloudTrail Precios. Para obtener información acerca de los precios de Amazon S3, consulte Precios de Amazon S3.

CloudTrail Almacenes de datos de eventos en Lake

CloudTrail Lake le permite ejecutar consultas basadas en SQL en sus eventos. CloudTrail Lake convierte los eventos existentes en formato JSON basado en filas al formato Apache ORC. ORC es un formato de almacenamiento en columnas optimizado para una recuperación rápida de datos. Los eventos se agregan en almacenes de datos de eventos, que son recopilaciones inmutables de eventos en función de criterios que se seleccionan aplicando selectores de eventos

avanzados. Los selectores que se aplican a un almacén de datos de eventos controlan los eventos que perduran y están disponibles para la consulta. Para obtener más información sobre CloudTrail Lake, consulte Cómo trabajar con AWS CloudTrail Lake en la Guía del AWS CloudTrail usuario.

CloudTrail Los almacenes de datos y las consultas sobre eventos de Lake conllevan costes. Cuando crea un almacén de datos de eventos, elige la opción de precios que desea utilizar para él. La opción de precios determina el costo de la incorporación y el almacenamiento de los eventos, así como el periodo de retención predeterminado y máximo del almacén de datos de eventos. Para obtener más información sobre CloudTrail los precios, consulte AWS CloudTrail Precios.

Eventos de datos en CloudTrail

Los eventos de datos proporcionan información sobre las operaciones de recursos realizadas en o dentro de un recurso (por ejemplo, leer o escribir en un objeto de Amazon S3). Se denominan también operaciones del plano de datos. Los eventos de datos suelen ser actividades de gran volumen. De forma predeterminada, CloudTrail no registra los eventos de datos. El historial de CloudTrail eventos no registra los eventos de datos.

Se aplican cargos adicionales a los eventos de datos. Para obtener más información sobre CloudTrail los precios, consulta AWS CloudTrail Precios.

Puede registrar eventos de datos para los tipos de Step Functions recursos mediante la CloudTrail consola o las operaciones de la CloudTrail API. AWS CLI Para obtener más información sobre cómo registrar los eventos de datos, consulte Registro de eventos de datos con la AWS Management Console y Registro de eventos de datos con la AWS Command Line Interface en la Guía del usuario de AWS CloudTrail .

En la siguiente tabla se enumeran los tipos de Step Functions recursos para los que puede registrar eventos de datos. La columna de tipos de eventos de datos muestra el valor que se puede elegir en la lista de tipos de eventos de datos de la CloudTrail consola. La columna de valores resources.type muestra el `resources.type` valor que se debe especificar al configurar los selectores de eventos avanzados mediante las API o. AWS CLI CloudTrail La CloudTrail columna API de datos en la que se ha registrado muestra las llamadas a la API registradas CloudTrail para el tipo de recurso.

Puede configurar selectores de eventos avanzados para filtrar según los campos `eventName`, `readOnly` y `resources.ARN` y así registrar solo los eventos que son importantes para usted. Para

obtener más información acerca de estos campos, consulte AdvancedFieldSelector en la Referencia de la API de AWS CloudTrail .

Tipo de evento de datos	resources.type value	API de datos registradas en CloudTrail
Máquina de estado de Step Functions	`AWS::StepFunctions::StateMachine`	• InvokeHTTPEndpoint

Eventos de gestión en CloudTrail

Los eventos de administración proporcionan información sobre las operaciones de administración que se llevan a cabo en los recursos de su empresa Cuenta de AWS. Se denominan también operaciones del plano de control. De forma predeterminada, CloudTrail registra los eventos de administración.

State Machine (Máquina de estado)

- CreateStateMachine
- ListStateMachines
- DescribeStateMachine
- UpdateStateMachine
- DeleteStateMachine
- ValidateStateMachineDefinition
- TestState

Alias de State Machine

- CreateStateMachineAlias
- ListStateMachineAliases
- DescribeStateMachineAlias
- UpdateStateMachineAlias
- DeleteStateMachineAlias

Versión de State Machine

- ListStateMachineVersions
- PublishStateMachineVersion
- DeleteStateMachineVersion

Ejecuciones

- StartExecution
- StartSyncExecution
- RedriveExecution
- ListExecutions
- DescribeExecution
- GetExecutionHistory
- DescribeStateMachineForExecution
- StopExecution

Actividad

- CreateActivity
- ListActivities
- DescribeActivity
- DeleteActivity
- GetActivityTask

Token de tarea

- SendTaskSuccess
- SendTaskHeartbeat
- SendTaskFailure

MapRun

- ListMapRuns
- DescribeMapRun

- UpdateMapRun

Etiquetas

- ListTagsForResource
- TagResource
- UntagResource

Ejemplos de evento

Un evento representa una solicitud única de cualquier fuente e incluye información sobre la operación de API solicitada, la fecha y la hora de la operación, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que los eventos no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra un evento de CloudTrail datos que lo demuestra`InvokeHTTPEndpoint`.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "states.amazonaws.com"
    },
    "eventTime": "2024-05-01T01:23:45Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "InvokeHTTPEndpoint",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "states.amazonaws.com",
    "userAgent": "states.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::StepFunctions::StateMachine",
            "ARN": "arn:aws:states:us-
east-1:123456789012:stateMachine:ExampleStateMachine"
```

```
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "serviceEventDetails": {
        "httpMethod": "GET",
        "httpEndpoint": "https://example.com"
    },
    "eventCategory": "Data"
}
```

El siguiente ejemplo muestra un evento CloudTrail de administración que demuestra la `CreateStateMachine` operación.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJYDLDBVBI4EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/test-user",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "test-user"
    },
    "eventTime": "2024-05-01T01:23:45Z",
    "eventSource": "states.amazonaws.com",
    "eventName": "CreateStateMachine",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "name": "MyStateMachine",
        "definition": "HIDDEN_DUE_TO_SECURITY_REASONS",
        "roleArn": "arn:aws:iam::123456789012:role/MyStateMachineRole",
        "type": "STANDARD",
        "loggingConfiguration": {
            "level": "OFF",
            "includeExecutionData": false
        },
        "tags": [],
        "tracingConfiguration": {
            "enabled": false
```

```
        },
        "publish": false
    },
    "responseElements": {
        "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:MyStateMachine",
        "creationDate": "May 1, 2024 1:23:45 AM"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


Para obtener información sobre el contenido de los CloudTrail registros, consulte el contenido de los CloudTrail registros en la Guía del AWS CloudTrail usuario.

Registro mediante CloudWatch Logs

Los flujos de trabajo estándar registran el historial de ejecuciones en AWS Step Functions, aunque opcionalmente puede configurar el registro en Registros de Amazon CloudWatch.

A diferencia de los flujos de trabajo estándar, los flujos de trabajo rápidos no registran el historial de ejecuciones en AWS Step Functions. Para consultar el historial de ejecuciones y los resultados de un flujo de trabajo rápido, tiene que configurar el registro en Registros de Amazon CloudWatch. La publicación de registros no bloquea ni ralentiza las ejecuciones.

> **Note**
>
> Al configurar el registro, se aplicarán los cargos de CloudWatch Logs y se le facturarán según la tarifa de los registros distribuidos. Para obtener más información, consulte Registros distribuidos en la pestaña Registros en la página Precios de CloudWatch.

Configuración de registros

Cuando crea un flujo de trabajo estándar mediante la consola de Step Functions, no se configurará para habilitar el registro en CloudWatch Logs. Un flujo de trabajo rápido creado mediante la consola

de Step Functions se configurará de forma predeterminada para habilitar el registro en CloudWatch Logs.

Para los flujos de trabajo rápidos, Step Functions puede crear un rol con la política de AWS Identity and Access Management (IAM) necesaria para CloudWatch Logs. Si se crea un flujo de trabajo estándar o un flujo de trabajo rápido mediante la API, la CLI o AWS CloudFormation, Step Functions no habilitará el registro de forma predeterminada y tendrá que asegurarse de que el rol tenga los permisos necesarios.

En cada ejecución que se inicia desde la consola, Step Functions proporciona un enlace a CloudWatch Logs, configurado con el filtro correcto para obtener los eventos de registro específicos para dicha ejecución.

Para configurar el registro, puede transferir el parámetro LoggingConfiguration al utilizar CreateStateMachine o UpdateStateMachine. Puede analizar aún más los datos en CloudWatch Logs mediante CloudWatch Logs Insights. Para obtener más información, consulte Análisis de los datos de registro con información de CloudWatch Logs.

Cargas de CloudWatch Logs

Los eventos del historial de ejecución pueden contener propiedades de entrada o salida en sus definiciones. Si la entrada o la salida de escape enviada a CloudWatch Logs supera los 248 KB, se truncará como resultado de las cuotas de CloudWatch Logs.

- Para determinar si una carga se ha truncado, revise las propiedades `inputDetails` y `outputDetails`. Para obtener más información, consulte el Tipo de datos `HistoryEventExecutionDataDetails`.
- En el caso de los flujos de trabajo estándar, puede ver el historial de ejecución completo mediante `GetExecutionHistory`.
- `GetExecutionHistory` no está disponible para flujos de trabajo rápidos. Puede usar los ARN de Amazon S3 para ver las entradas y salidas completas. Para obtener más información, consulte the section called “Utilizar los ARN de Amazon S3 en lugar de pasar cargas de gran tamaño”.

Políticas de IAM para registrarse en CloudWatch Logs

También tendrá que configurar el rol de IAM de ejecución de su máquina de estado para que tenga el permiso adecuado para registrarse en CloudWatch Logs, como se muestra en el siguiente ejemplo.

Ejemplo de políticas de IAM

A continuación se muestra una política de ejemplo que puede utilizar para configurar los permisos. Como se muestra en el siguiente ejemplo, debe especificar * en el campo `Resource` porque las acciones de la API de CloudWatch, como CreateLogDelivery y DescribeLogGroups, no admiten Tipos de recursos definidos por Amazon CloudWatch Logs. Para obtener más información, consulte Acciones definidas por Amazon CloudWatch Logs.

- Para obtener información sobre los recursos de CloudWatch, consulte Recursos y operaciones de CloudWatch Logs en la Guía del usuario de Amazon CloudWatch.
- Para obtener información sobre los permisos necesarios para configurar el envío de registros a CloudWatch Logs, consulte Permisos de usuario en la sección titulada Registros enviados a 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": "*"
        }
    ]
}
```

No se puede acceder a los CloudWatch Logs

Si no es posible acceder a los CloudWatch Logs, asegúrese de hacer lo siguiente:

1. Configurar el rol de IAM de ejecución de su máquina de estado para que tenga el permiso adecuado para registrarse en CloudWatch Logs.

Si utiliza las solicitudes CreateStateMachine o UpdateStateMachine, asegúrese de haber especificado el rol de IAM en el parámetro `roleArn` que contiene los permisos, como se muestra en el ejemplo anterior.

2. Comprobar que la política de recursos de CloudWatch Logs no supere el límite de 5120 caracteres.

   Si se ha superado el límite de caracteres, elimine los permisos innecesarios de la política de recursos de CloudWatch Logs o ponga el prefijo al nombre del grupo de registros con `/aws/vendedlogs`, lo que concederá permisos al grupo de registros sin añadir más caracteres a la política de recursos. Al crear un grupo de registros en la consola de Step Functions, a los nombres de los grupos de registro se les añade el prefijo `/aws/vendedlogs/states`. Para obtener más información, consulte Restricciones de tamaño de política de recursos de registros de Amazon CloudWatch.

Niveles de registro

Puede elegir entre `OFF`, `ALL`, `ERROR` o `FATAL`. No se registra ningún tipo de evento cuando se establece en `OFF` y se registran todos los tipos de eventos cuando se establece en `ALL`. Para `ERROR` y `FATAL`, consulte la tabla siguiente.

Para obtener más información sobre los datos de ejecución que se muestran para las ejecuciones de flujos de trabajo rápidos en función de estos Niveles de registro, consulte Ejecuciones de flujos de trabajo estándar y rápidos en la consola.

Tipo de evento	**ALL**	**ERROR**	**FATAL**	**OFF**
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
Execution Aborted	✓	✓	✓	
ExecutionFailed	✓	✓	✓	

Tipo de evento	ALL	ERROR	FATAL	OFF
ExecutionStarted	✓			
Execution Succeeded	✓			
Execution TimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		
MapIterationFailed	✓	✓		

Tipo de evento	ALL	ERROR	FATAL	OFF
MapIterationStarted	✓			
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			
ParallelStateFailed	✓	✓		

Tipo de evento	ALL	ERROR	FATAL	OFF
ParallelStateStarted	✓			
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		
TaskStateAborted	✓	✓		
TaskStateEntered	✓			
TaskStateExited	✓			
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			

Tipo de evento	ALL	ERROR	FATAL	OFF
TaskTimedOut	✓	✓		
WaitState Aborted	✓	✓		
WaitState Entered	✓			
WaitStateExited	✓			

AWS X-Ray y Step Functions

Puede utilizar AWS X-Ray para visualizar los componentes de su máquina de estado, identificar cuellos de botella en el rendimiento y solucionar problemas de solicitudes que dieron lugar a un error. Su máquina de estado envía datos de rastreo a X-Ray, y X-Ray procesa los datos para generar un mapa de servicio y resúmenes de rastreo con capacidad de búsqueda.

Con X-Ray activado para su máquina de estados, puede rastrear las solicitudes a medida que se ejecutan en Step Functions, en todas AWS las regiones donde X-Ray esté disponible. Esto le aporta información general detallada de una solicitud completa de Step Functions. Step Functions enviará registros de seguimiento a X-Ray para ejecuciones de máquinas de estado, incluso cuando un ID de seguimiento no se pasa por un servicio ascendente. Puede usar un mapa del servicio de X-Ray para ver la latencia de una solicitud, incluidos los AWS servicios que estén integrados con X-Ray. También puede configurar reglas de muestreo para indicar a X-Ray qué solicitudes debe registrar y a qué velocidad de muestreo, de acuerdo con los criterios que especifique.

Cuando X-Ray no está habilitado para su máquina de estado y un servicio ascendente no transfiere un ID de registro de seguimiento, Step Functions no enviará registros de seguimiento a X-Ray para ejecuciones de máquinas de estado. Sin embargo, si un servicio ascendente transfiere un ID de seguimiento, Step Functions enviará registros de seguimiento a X-Ray para ejecuciones de máquinas de estado.

Se puede utilizar AWS X-Ray con Step Functions en las regiones en las que se admiten ambas funciones. Consulte las páginas de puntos de conexión y cuotas de Step Functions y X-Ray para obtener información sobre la compatibilidad con regiones de X-Ray y Step Functions.

Cuotas combinadas de X-Ray y Step Functions

Puede añadir datos a un registro de seguimiento durante un máximo de siete días y consultar datos de rastreo que se remonten a treinta días, que es el tiempo que X-Ray almacena los datos de rastreo. Los registros de seguimiento estarán sujetos a las cuotas de X-Ray. Además de otras cuotas, X-Ray proporciona un tamaño de seguimiento mínimo garantizado de 100 KB para las máquinas de estado de Step Functions. Si se proporcionan más de 100 KB de datos de rastreo a X-Ray, esto puede dar lugar a un rastreo congelado. Consulte la sección de cuotas de servicio de la página Puntos de conexión y cuotas de X-Ray para obtener más información sobre otras cuotas de X-Ray.

Important

Step Functions no admite el seguimiento de X-Ray para las ejecuciones de flujos de trabajo secundarios iniciadas por un estado Distributed Map, ya que el límite de tamaño de los documentos de rastreo se supera con facilidad para dichas ejecuciones.

Temas

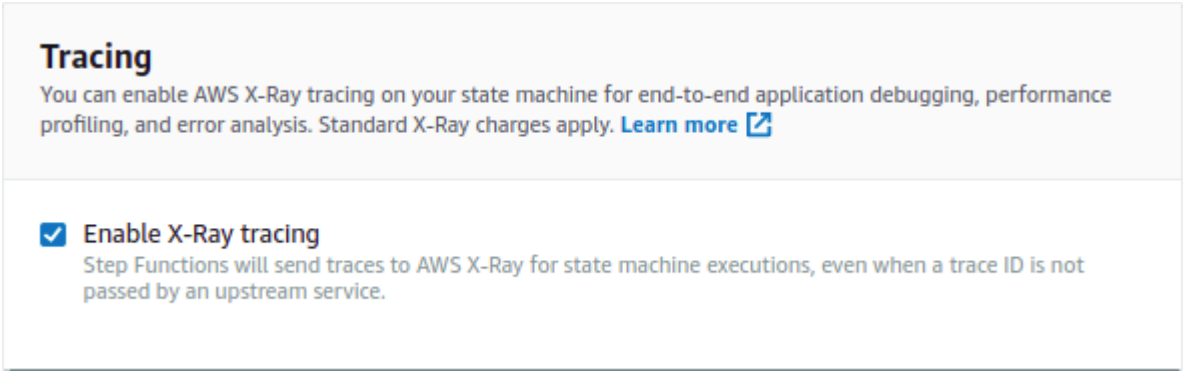
Ajustes y configuración

Habilitar el rastreo de X-Ray al crear una máquina de estado

Seleccione Habilitar rastreo de X-Ray en la página Especificar detalles para habilitar el rastreo de X-Ray al crear una nueva máquina de estado.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En la página Elegir un método de creación, seleccione la opción adecuada para crear su máquina de estado. Si se selecciona Ejecutar un proyecto de muestra, no se podrá habilitar el rastreo de X-Ray durante la creación de la máquina de estado y tendrá que habilitarse una vez creada la máquina de estado. Para obtener más información sobre cómo habilitar X-Ray en una máquina de estado existente, consulte Habilitar X-Ray en una máquina de estado existente.

   Elija Siguiente.
3. Configure su máquina de estado en la página Especificar detalles.
4. Seleccione Habilitar rastreo de X-Ray.



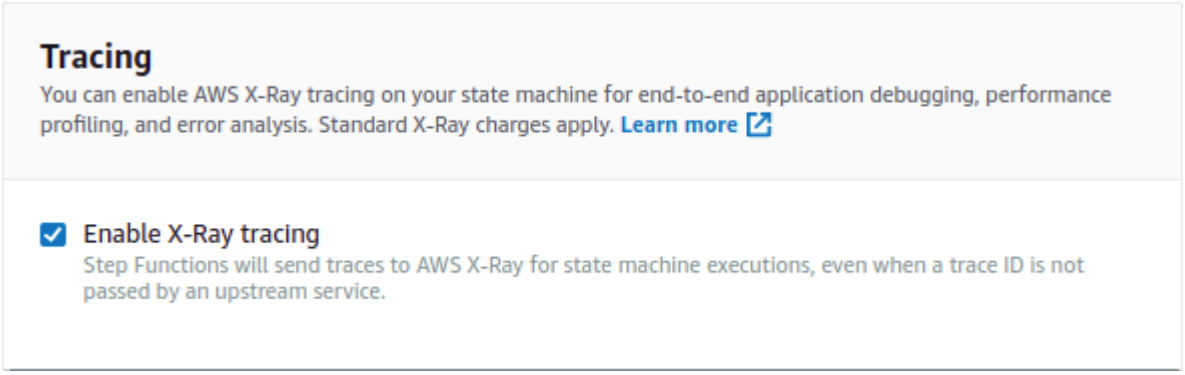
La máquina de estado de Step Functions enviará ahora registros de seguimiento a X-Ray para las ejecuciones de máquinas de estado.

> **Note**
>
> Si decide usar un rol de IAM existente, debe asegurarse de que se permiten las escrituras de X-Ray. Para obtener más información sobre los permisos necesarios, consulte las Políticas de IAM para X-Ray.

Habilitar X-Ray en una máquina de estado existente

Para habilitar X-Ray en una máquina de estado existente:

1. En la consola de Step Functions, seleccione la máquina de estado para la que desee activar el rastreo.
2. Elija Editar.
3. Seleccione Habilitar rastreo de X-Ray.



   Aparecerá una notificación en la que se indicará que es posible que haya que hacer cambios adicionales.

> **Note**
>
> Al habilitar X-Ray para una máquina de estado existente, debe asegurarse de tener una política de IAM que otorgue permisos suficientes para que X-Ray realice rastreos. Puede añadir una manualmente o generarla. Para obtener más información, consulte la sección Política de IAM para Políticas de IAM para AWS X-Ray.

4. (Opcional) Genere automáticamente un nuevo rol para su máquina de estado para incluir los permisos de X-Ray.
5. Seleccione Guardar.

Configurar el rastreo de X-Ray para Step Functions

La primera vez que ejecute una máquina de estados con el rastreo de rayos X activado, utilizará los valores de configuración predeterminados para el rastreo de rayos X. AWS X-Ray no recopila datos de cada solicitud que se envía a una aplicación. En su lugar, recopila datos para un número estadísticamente significativo de solicitudes. De forma predeterminada, se registra la primera

solicitud cada segundo y el cinco por ciento de las solicitudes adicionales. Una petición por segundo es el depósito. Esto garantiza que se registre al menos un registro de seguimiento cada segundo mientras el servicio atienda solicitudes. El cinco por ciento es el porcentaje al que se muestrean las solicitudes adicionales más allá del tamaño del depósito.

Para evitar incurrir en cargos por servicio al empezar, la tasa de muestreo predeterminada es conservadora. Puede configurar X-Ray para modificar la regla de muestreo predeterminada y configurar reglas adicionales que aplican el muestreo en función de las propiedades del servicio o solicitud.

Por ejemplo, es posible que desee deshabilitar el muestreo y rastrear todas las solicitudes de llamadas que modifiquen el estado o el manejo Cuentas de AWS de las transacciones. Para llamadas de solo lectura de alto volumen, como sondeo en segundo plano, comprobaciones de estado o mantenimiento de conexión, es posible muestrear con un bajo índice y seguir obteniendo datos suficientes para ver los problemas que se produzcan.

Para configurar una regla de muestreo para su máquina de estado:

1. Vaya a la consola de X-Ray.
2. Seleccione Muestreo.
3. Para crear una regla, elija Crear regla de muestreo.

   Para editar una regla, elija el nombre de una regla.

   Para eliminar una regla, elija una regla y utilice el menú Acciones para eliminarla.

Algunas partes de las reglas de muestreo existentes, como el nombre y la prioridad, no se pueden cambiar. En su lugar, es posible añadir o clonar una regla existente, realizar los cambios que se deseen y, posteriormente, utilizar la nueva regla.

Para obtener información detallada sobre las reglas de muestreo de X-Ray y cómo configurar los distintos parámetros, consulte Configuración de reglas de muestreo en la consola de X-Ray.

Integrar los servicios ascendentes

Para integrar la ejecución de los flujos de trabajo de Step Functions, como los flujos de trabajo rápidos, sincrónico y estándar, con un servicio ascendente, debe configurar el `traceHeader`. Esto se hace automáticamente si utiliza una API de HTTP en API Gateway. Sin embargo, si utiliza una

función de Lambda o un SDK, tendrá que configurar por sí mismo el `traceHeader` en las llamadas a la API `StartExecution` o `StartSyncExecution`.

Debe especificar el formato de `traceHeader` como `\p{ASCII}#`. Además, para que Step Functions pueda usar el mismo ID de seguimiento, debe especificar el formato como `Root={TRACE_ID};Sampled={1 or 0}`. Si utiliza una función de Lambda, reemplace `TRACE_ID` por el ID de seguimiento del segmento actual y establezca el campo muestreado como `1` si el modo de muestreo fuera verdadero y `0` si el modo de muestreo fuera falso. Si se proporciona el ID de seguimiento en este formato, obtendrá un seguimiento completo.

A continuación se muestra un ejemplo escrito en Python para mostrar cómo especificar el `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)
```

Conceptos

La consola de X-Ray

La AWS X-Ray consola le permite ver los mapas de servicio y los rastreos de las solicitudes que atienden sus aplicaciones. Puede acceder a la consola para ver la información detallada recopilada por X-Ray cuando está habilitado en su máquina de estados.

Consulte Visualización de la consola de X-Ray para obtener información sobre cómo acceder a la consola de X-Ray para las ejecuciones de máquinas de estado.

Para obtener información detallada sobre la consola X-Ray, consulte la documentación de la consola de X-Ray.

Segmentos, subsegmentos y registros de seguimiento

Un segmento registra información sobre una solicitud enviada a su máquina de estado. Contiene información como el trabajo que realiza la máquina de estado y también puede contener subsegmentos con información sobre las llamadas posteriores.

Un registro de seguimiento recopila todos los segmentos que genera una solicitud.

Muestreo

Para garantizar un rastreo eficaz y proporcionar una muestra representativa de las solicitudes que su aplicación atiende, X-Ray aplica un algoritmo de muestreo para determinar qué solicitudes se rastrearán. Esto se puede cambiar editando las reglas de muestreo.

Métricas

Para su máquina de estado, X-Ray medirá el tiempo de invocación, el tiempo de transición de estado, el tiempo total de ejecución de Step Functions y las variaciones en este tiempo de ejecución. Se puede acceder a esta información a través de la consola de X-Ray.

Análisis

La consola de AWS X-Ray Analytics es una herramienta interactiva para interpretar los datos de rastreo. Puede acotar el conjunto de datos con filtros cada vez más detallados haciendo clic en los gráficos y los paneles de métricas y campos asociados al conjunto de registros de seguimiento actual. Esto le permite analizar el rendimiento de su máquina de estado y localizar e identificar rápidamente los problemas de rendimiento.

Para obtener información detallada sobre los análisis de X-Ray, consulte Interactuar con la consola de AWS X-Ray Analytics

Integración de los servicios de Step Functions y X-Ray

Algunos de los AWS servicios que se integran con Step Functions permiten la integración AWS X-Ray añadiendo un encabezado de rastreo a las solicitudes, ejecutando el daemon X-Ray o tomando

decisiones de muestreo y cargando los datos de rastreo a X-Ray. Otros deben instrumentarse mediante el SDK. AWS X-Ray Algunos aún no admiten la integración de X-Ray. La integración de X-Ray es necesaria para proporcionar datos de rastreo completos cuando se utiliza una integración de servicios con Step Functions

Soporte nativo de X-Ray

Las integraciones de servicios con soporte nativo de X-Ray incluyen:

- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS Lambda
- AWS Step Functions

Instrumentación requerida

Integraciones de servicios que requieren la instrumentación X-Ray:

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

Rastreo solo del lado del cliente

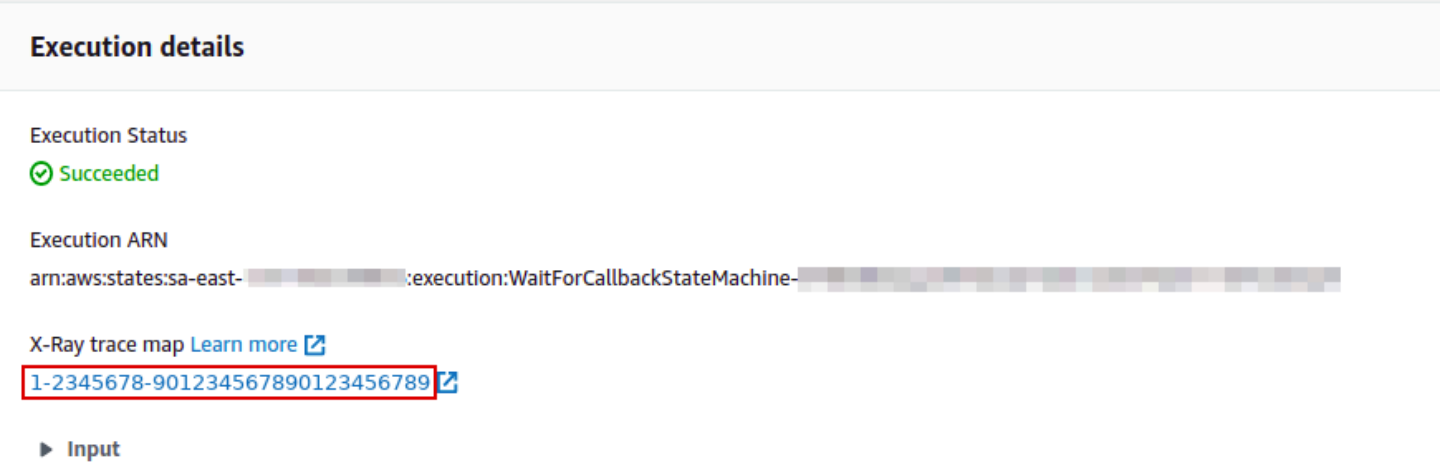
Otras integraciones de servicios no admiten registros de seguimiento de X-Ray. Sin embargo, aún se pueden recopilar rastreos del lado del cliente:

- Amazon DynamoDB
- Amazon EMR
- Amazon SageMaker
- AWS CodeBuild
- AWS Glue

Visualización de la consola de X-Ray

Los servicios envían datos a X-Ray como segmentos. X-Ray agrupa los segmentos que tienen una solicitud en común en registros de seguimiento. X-Ray procesa los registros de seguimiento para generar un gráfico de servicios, que constituye una representación visual de la aplicación.

Una vez iniciada la ejecución de la máquina de estado, se pueden ver los registros de seguimiento de X-Ray al seleccionar el enlace al mapa de registros de seguimiento de X-Ray en la sección Detalles de ejecución.



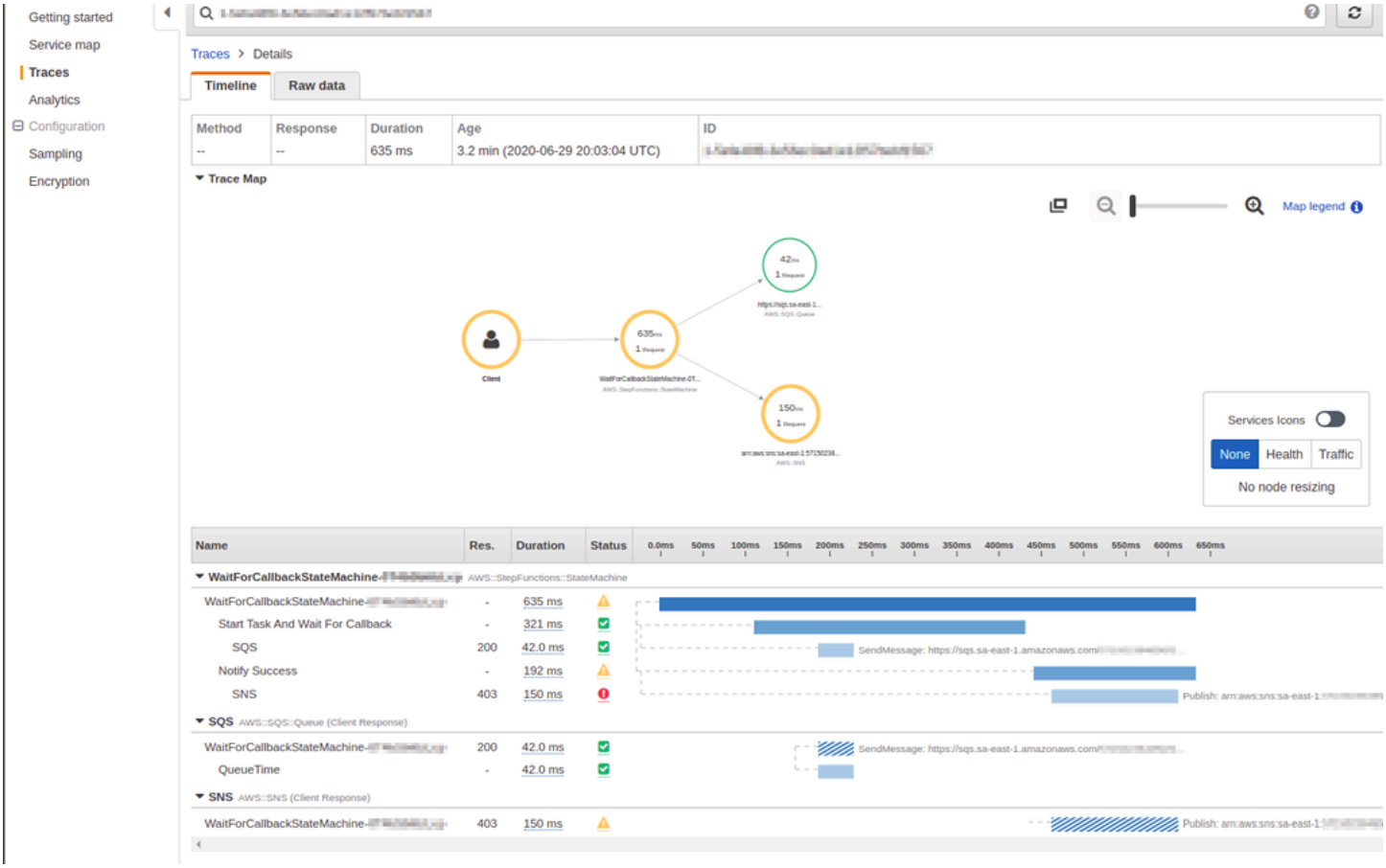
Una vez que haya habilitado X-Ray en la máquina de estado, puede ver la información de rastreo de sus ejecuciones en la consola de X-Ray.

Visualización de la información de rastreo de X-Ray para Step Functions

Los siguientes pasos ilustran el tipo de información que puede ver en la consola después de habilitar X-Ray y comenzar una ejecución. Se muestran registros de seguimiento de X-Ray para el proyecto de muestra Ejemplo de patrón de devolución de llamadas (Amazon SQS, Amazon SNS, Lambda) .

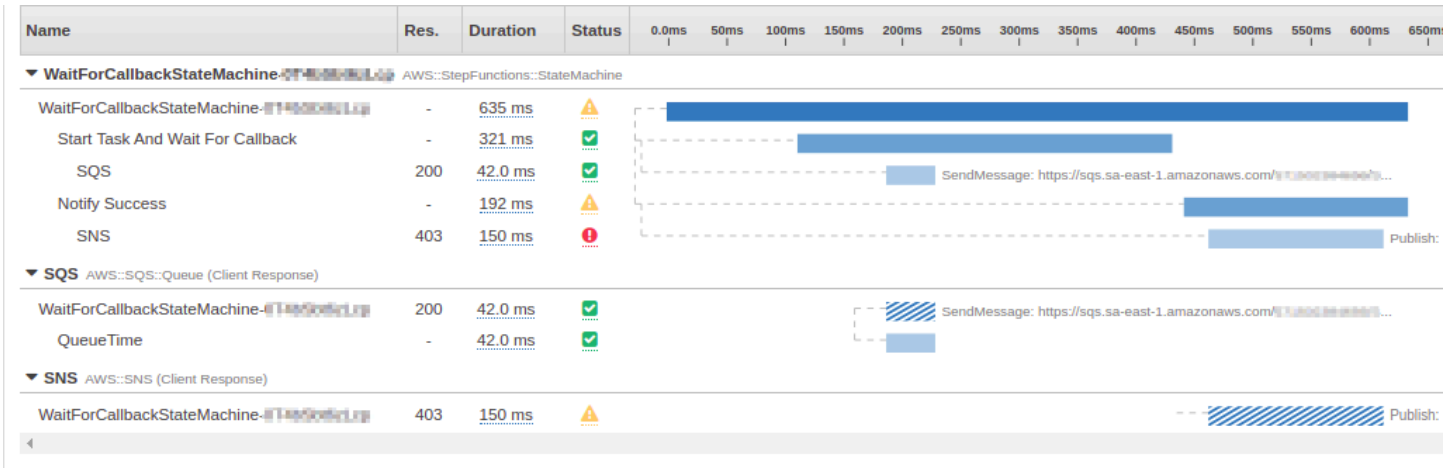
Registros de seguimiento

Una vez finalizada la ejecución, puede ir a la consola de X-Ray, donde aparecerá la página Registros de seguimiento de X-Ray. Aquí se muestra información general del mapa de servicio, así como información de registros de seguimiento y segmentos de la máquina de estado.

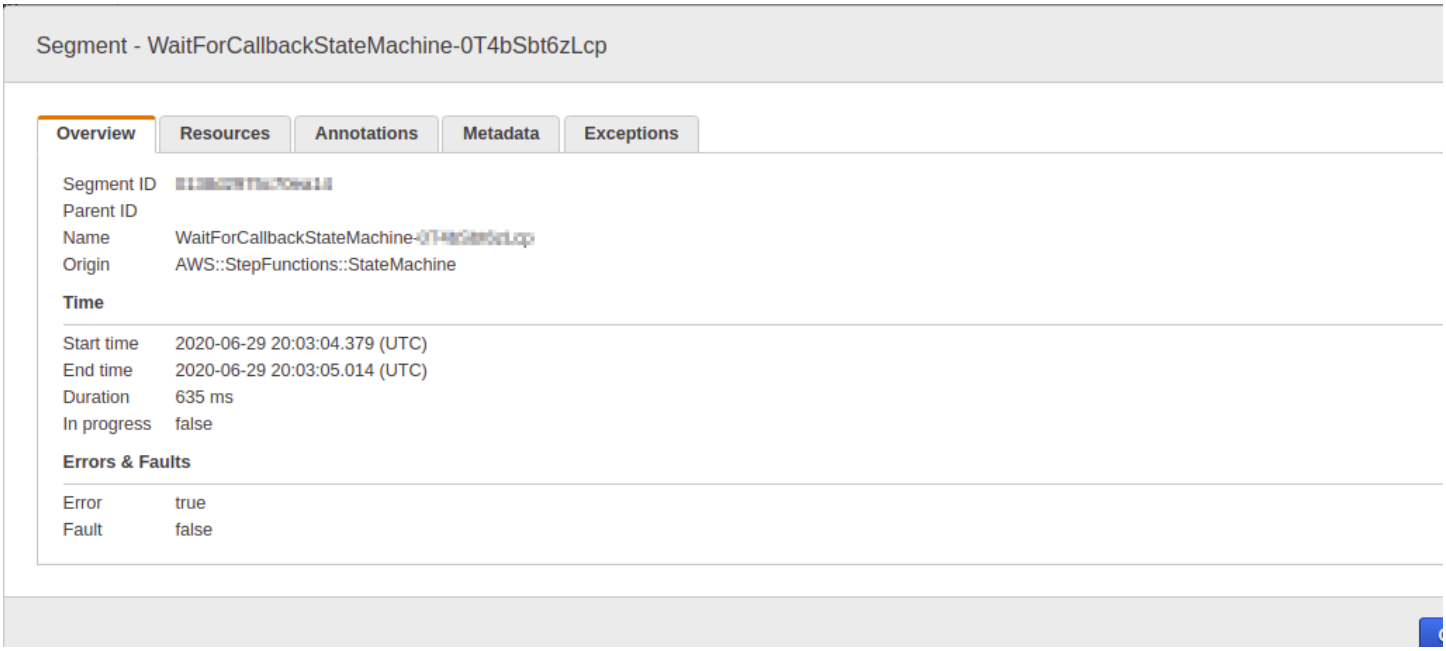


Mapa de servicios

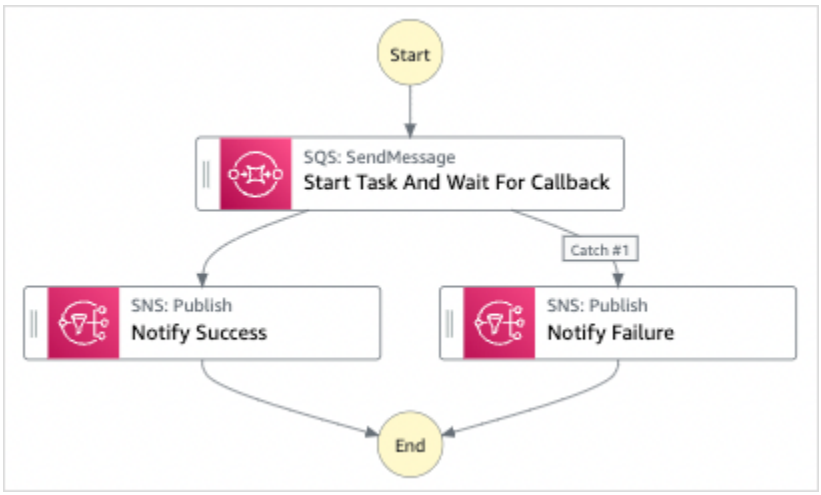
El mapa de servicio en la consola de X-Ray le ayuda a identificar los servicios donde ocurran errores, donde haya conexiones con alta latencia o donde se observen registros de seguimiento de solicitudes que dieron error.



En el mapa de registros de seguimiento, puede elegir un nodo de servicio para ver sus solicitudes o un límite entre dos nodos para ver las solicitudes que pasaron por esa conexión. Aquí, se ha seleccionado el nodo `WaitForCallBack` y puede ver información adicional sobre su estado de ejecución y respuesta.



Puede ver cómo se correlaciona el mapa del servicio de X-Ray con la máquina de estado. Hay un nodo de mapa de servicio para cada integración de servicios que es llamada por Step Functions, siempre que admita X-Ray.



Segmentos y subsegmentos

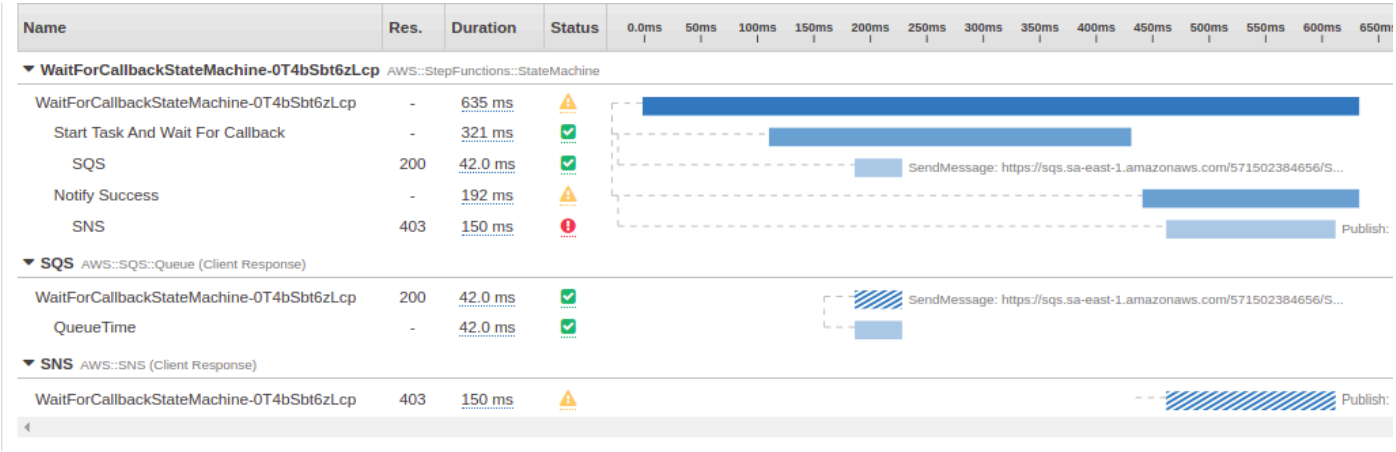
Un registro de seguimiento es una colección de segmentos que genera una solicitud. Cada segmento incluye el nombre del recurso, detalles de la solicitud y detalles del trabajo realizado.

En la página Registros de seguimiento, puede ver los segmentos y, si están expandidos, sus subsegmentos correspondientes. Puede elegir un segmento o subsegmento para ver información detallada sobre este.

Seleccione cada una de las pestañas para ver cómo se muestra la información de los segmentos y los subsegmentos.

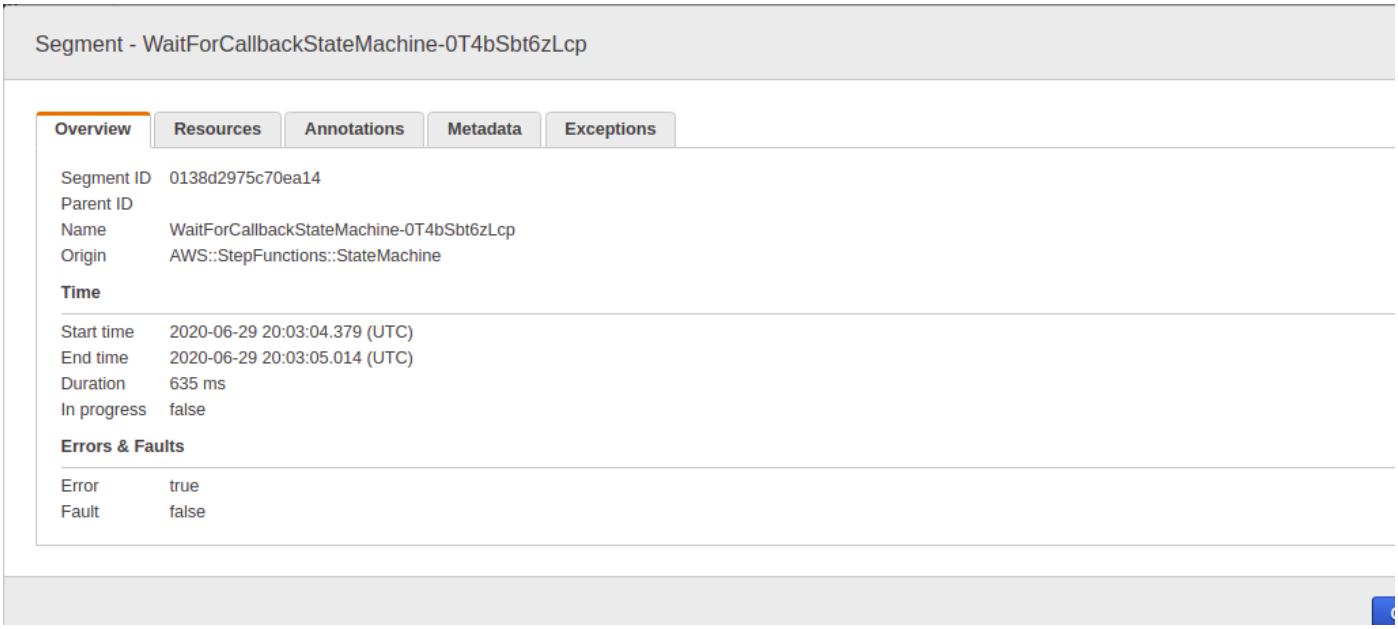
Overview of Segments

Información general de los segmentos y subsegmentos de esta máquina de estado. Hay un segmento diferente para cada nodo en el mapa de servicio.



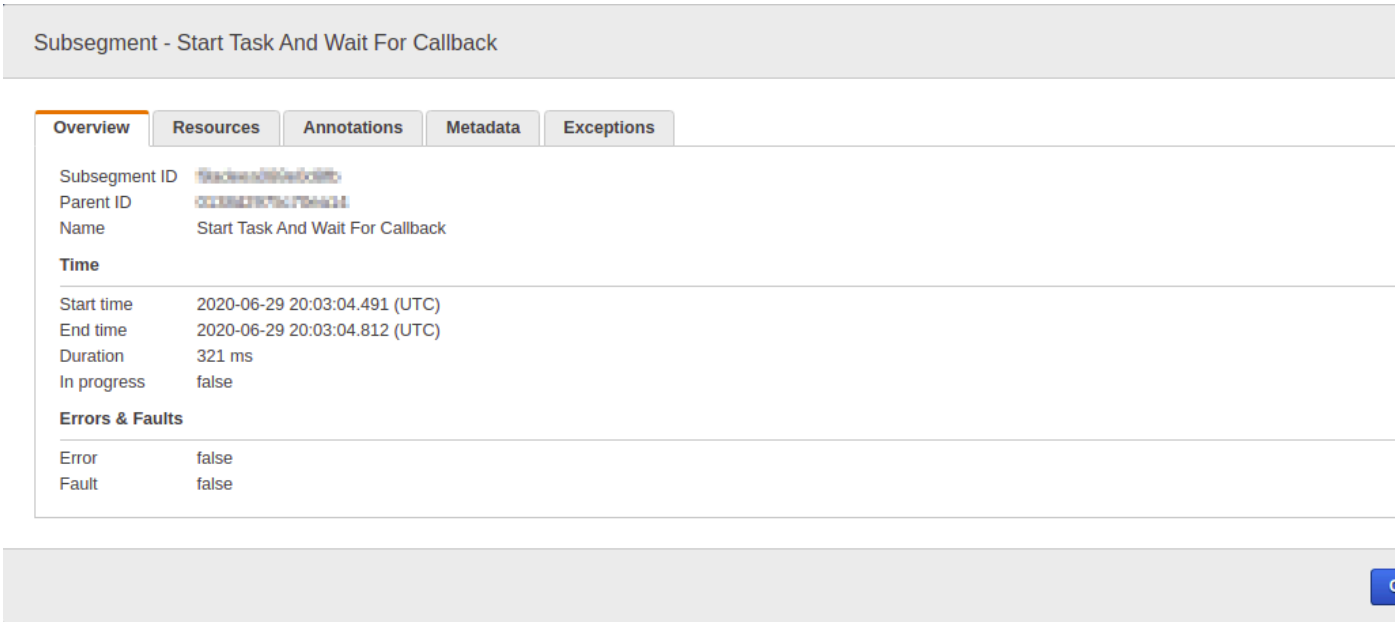
View segment detail

La selección de un segmento incluye el nombre del recurso, detalles de la solicitud y detalles del trabajo realizado.



View subsegment detail

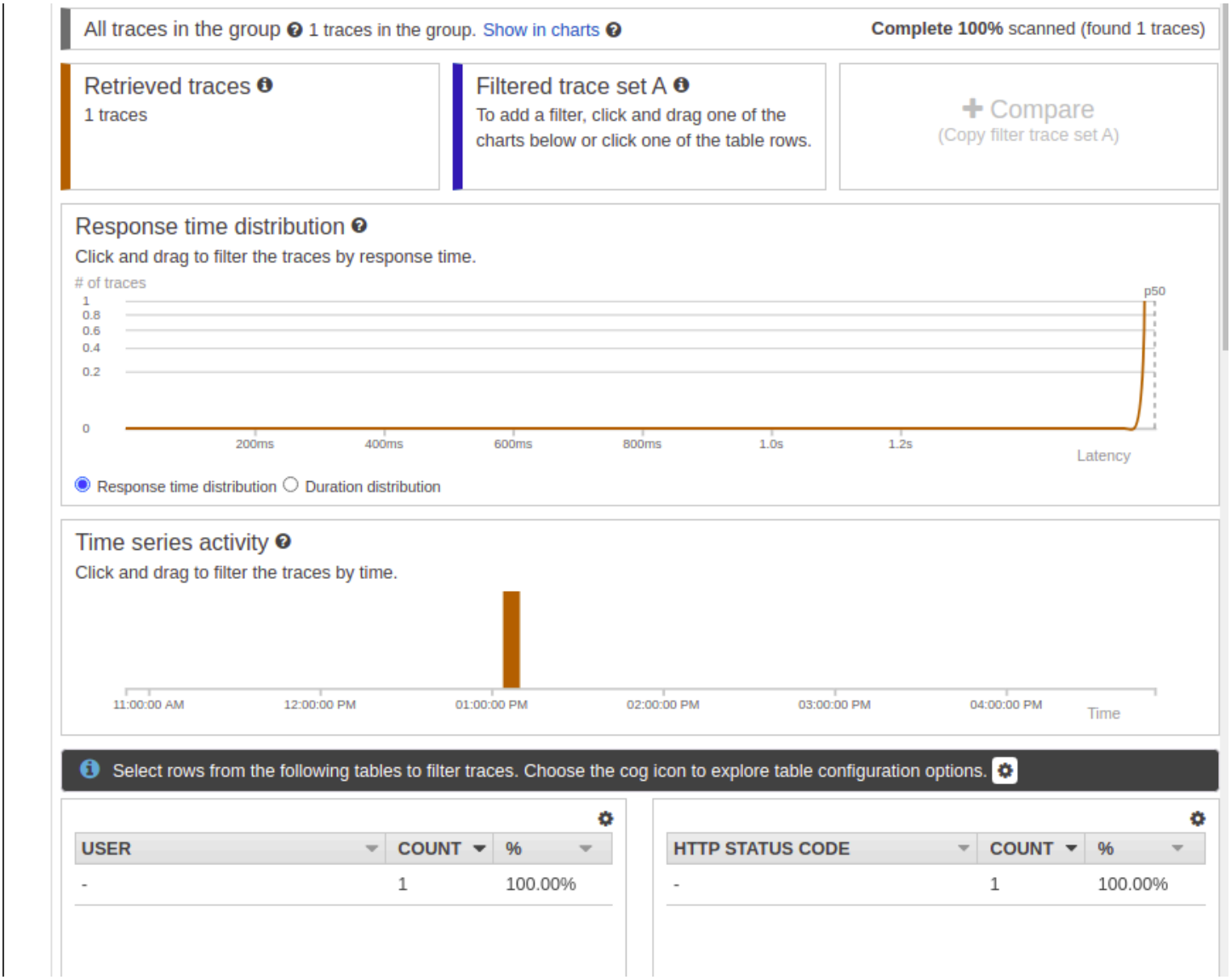
Un segmento puede desglosar los datos del trabajo realizado en subsegmentos. Al elegir un subsegmento, podrá ver información y detalles más precisos sobre la temporización. Un subsegmento puede contener detalles adicionales sobre una llamada a un AWS servicio, una API HTTP externa o una base de datos SQL.



Análisis

La consola AWS X-Ray de Analytics es una herramienta interactiva para interpretar los datos de rastreo. Puede utilizarla para comprender con más facilidad el rendimiento de su máquina de estado. La consola le permite explorar, analizar y visualizar los registros de seguimiento a través de gráficos de tiempo de respuesta y series temporales. Esto puede ayudarle a localizar rápidamente los problemas de rendimiento y latencia.

Puede acotar el conjunto de datos con filtros cada vez más detallados haciendo clic en los gráficos y los paneles de métricas y campos asociados al conjunto de registros de seguimiento actual.

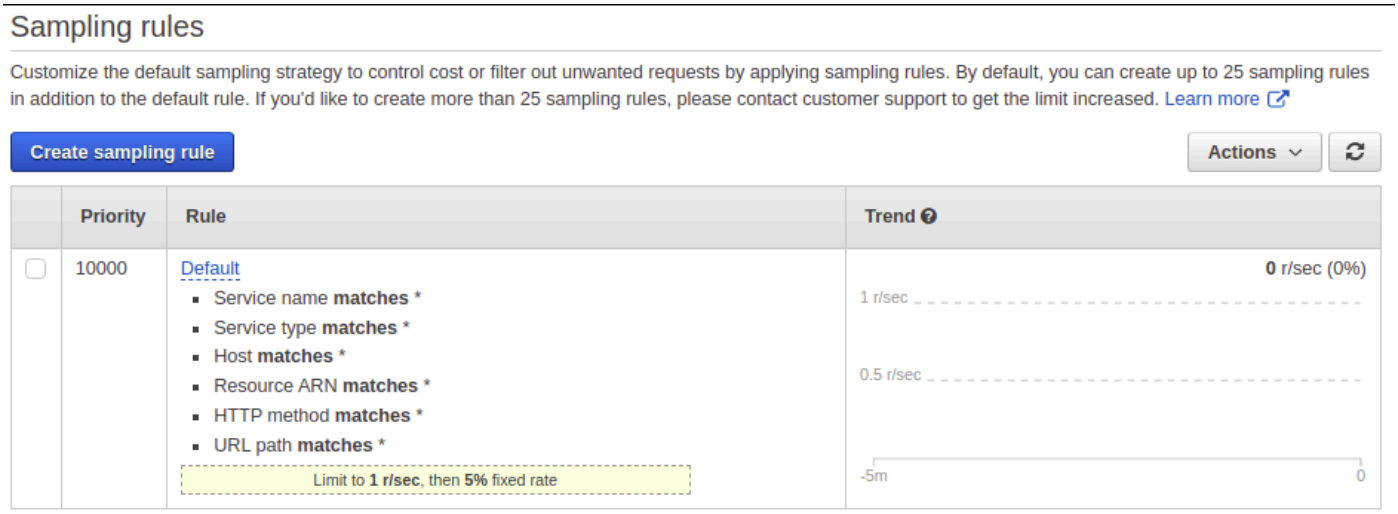


Configuración

Puede configurar las opciones de muestreo y cifrado desde la consola de X-Ray.

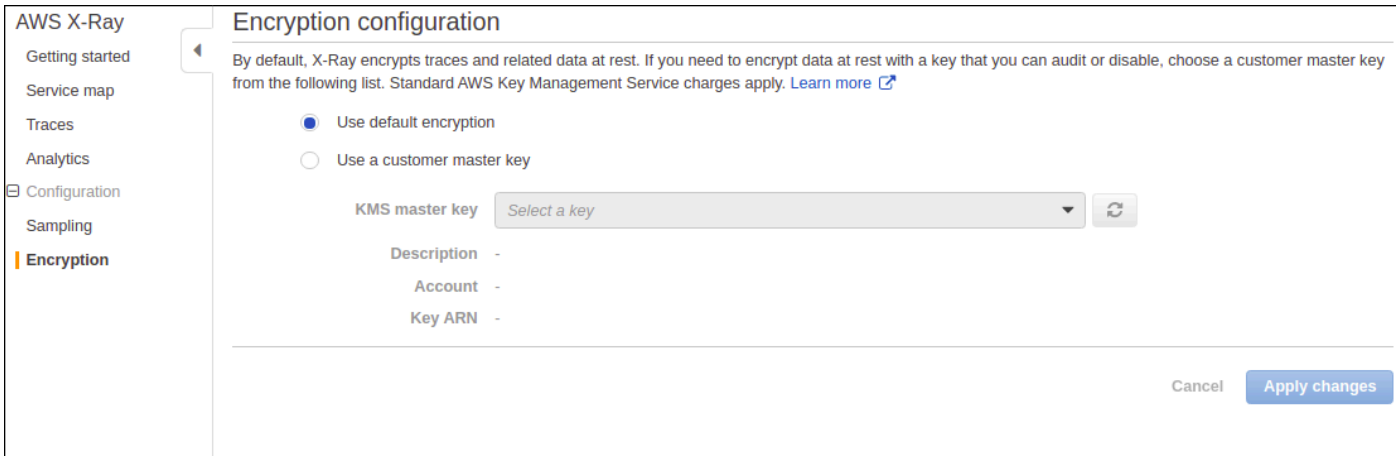
Sampling

Seleccione Muestreo para ver los detalles sobre la frecuencia y la configuración de muestreo. Puede cambiar las reglas de muestreo para controlar la cantidad de datos que se registran y modificar el comportamiento de muestreo para adaptarlo a sus requisitos específicos.



Encryption

Seleccione Cifrado para modificar la configuración de cifrado. Puede usar la configuración predeterminada, donde X-Ray cifra los registros de seguimiento y la fecha en reposo. Además, si es necesario, puede elegir una clave maestra de cliente. En este último caso, se aplican tasas de AWS KMS estándar.



¿Qué ocurre si no hay datos en el mapa de registros de seguimiento o en el mapa de servicio?

Si ha habilitado X-Ray, pero no ve ningún dato en la consola de X-Ray, compruebe que:

- Los roles de IAM están configurados correctamente para poder escribir en X-Ray.
- Las reglas de muestreo permiten el muestreo de datos.

- Dado que puede que haya un pequeño retraso hasta que se apliquen los roles de IAM recién creados o modificados, vuelva a comprobar los mapas de registros de seguimiento o de servicio transcurridos unos minutos.
- Si ve Datos no encontrados en el panel X-Ray Traces, compruebe la configuración de su cuenta de IAM y asegúrese de que AWS Security Token Service esté habilitada para la región deseada. Para obtener más información, consulte Activación y desactivación AWS STS en y en la Guía del Región de AWS usuario de IAM.

Uso de AWS User Notifications con AWS Step Functions

Puede usar AWS User Notifications para configurar los canales de entrega a fin de recibir notificaciones sobre los eventos de AWS Step Functions. Recibirá una notificación cuando un evento coincida con una regla que especifique. Puede recibir notificaciones de eventos a través de varios canales, como correo electrónico, notificaciones por chat de AWS Chatbot o notificaciones de inserción de AWS Console Mobile Application. También puede ver las notificaciones en el Centro de notificaciones de la consola. Las Notificaciones de usuario admiten la agregación, lo que puede reducir el número de notificaciones que recibe durante eventos específicos.

Seguridad en AWS Step Functions

En esta sección se proporciona información sobre AWS Step Functions la seguridad y la autenticación.

Step Functions utiliza la IAM para controlar el acceso a otros AWS servicios y recursos. Para obtener información general sobre cómo funciona IAM, consulte Información general sobre la administración del acceso en la Guía del usuario de IAM. Para obtener información general de credenciales de seguridad, consulte Credenciales de seguridad de AWS en la Referencia general de Amazon Web Services.

Protección de datos en AWS Step Functions

El modelo de responsabilidad AWS compartida modelo se aplica a la protección de datos en AWS Step Functions. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta todos los Nube de AWS. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Usted también es responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utilice la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.

- Utilice servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-2 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como, por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabajas con Step Functions u otras funciones Servicios de AWS mediante la consola, la API o AWS los SDK. AWS CLI Cualquier dato que ingrese en etiquetas o campos de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cifrado en AWS Step Functions

Cifrado en reposo

Step Functions siempre cifra sus datos en reposo. Los datos en reposo AWS Step Functions se cifran mediante un cifrado transparente del lado del servidor. Esto ayuda a reducir la carga y la complejidad operativas que conlleva la protección de información confidencial. Con el cifrado en reposo, puede crear aplicaciones sensibles a la seguridad que cumplen los requisitos de cifrado y normativos.

Cifrado en tránsito

Step Functions cifra los datos en tránsito entre el servicio y otros servicios de AWS integrados (consulte Uso AWS Step Functions con otros servicios). Todos los datos que pasan entre los servicios integrados y Step Functions se cifran mediante Transport Layer Security (TLS).

Administración de identidad y acceso en AWS Step Functions

El acceso a AWS Step Functions requiere credenciales que AWS pueda utilizar para autenticar sus solicitudes. Estas credenciales deben tener permisos para acceder a AWS los recursos, por ejemplo, para recuperar datos de eventos de otros AWS recursos. En las secciones siguientes, se incluye

información detallada sobre el uso de AWS Identity and Access Management (IAM) y Step Functions para ayudar a proteger sus recursos controlando quién puede acceder a ellos.

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. Step Functions La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo en el que se realice. Step Functions

Usuario del servicio: si utiliza el Step Functions servicio para realizar su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más Step Functions funciones para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en Step Functions, consulte Solución de problemas de AWS Step Functions identidad y acceso.

Administrador de servicios: si estás a cargo de Step Functions los recursos de tu empresa, es probable que tengas acceso total a ellos Step Functions. Su trabajo consiste en determinar a qué Step Functions funciones y recursos deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar la IAM Step Functions, consulte¿Cómo AWS Step Functions funciona con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a AWS. Para ver ejemplos de políticas Step Functions basadas en la identidad que puede utilizar en IAM, consulte. Ejemplos de políticas basadas en la identidad para AWS Step Functions

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (IAM Identity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, asumes un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWSen la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM, o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede

asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para conocer la diferencia entre las funciones y las políticas basadas en recursos para el acceso entre cuentas, consulte el tema sobre el acceso a los recursos entre cuentas en IAM en la Guía del usuario de IAM.
- Acceso entre servicios: algunos utilizan funciones en otros. Servicios de AWS Servicios de AWS Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para

obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

- Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations .
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en

función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Control de acceso

Aunque disponga de credenciales válidas para autenticar las solicitudes, si no tiene permisos, no podrá crear recursos de Step Functions ni obtener acceso a ellos. Por ejemplo, debe tener permisos para AWS Lambda invocar los destinos de Amazon Simple Notification Service (Amazon SNS) y Amazon Simple Queue Service (Amazon SQS) asociados a sus reglas de Step Functions.

En las secciones siguientes, se describe cómo administrar los permisos de Step Functions.

- Creación de un rol de IAM para la máquina de estado
- Creación de permisos granulares de IAM para usuarios no administradores
- Puntos de conexión de VPC para Step Functions
- Políticas de IAM para servicios integrados
- Políticas de IAM para usar el estado Map Distributed

Acciones políticas para Step Functions

Admite acciones de política	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no

tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de Step Functions acciones, consulta los recursos definidos por AWS Step Functions en la referencia de autorización del servicio.

Las acciones de política Step Functions utilizan el siguiente prefijo antes de la acción:

```
states
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "states:action1",
      "states:action2"
         ]
```

Para ver ejemplos de políticas Step Functions basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Step Functions

Recursos de políticas para Step Functions

Admite recursos de políticas	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de Step Functions recursos y sus ARN, consulte las acciones definidas por AWS Step Functions en la Referencia de autorización de servicios. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Recursos definidos por AWS Step Functions.

Para ver ejemplos de políticas Step Functions basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Step Functions

Claves de condición de la política para Step Functions

Admite claves de condición de políticas específicas del servicio	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de claves de Step Functions condición, consulte las claves de condición AWS Step Functions en la Referencia de autorización de servicio. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Recursos definidos por AWS Step Functions.

Para ver ejemplos de políticas Step Functions basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Step Functions

ACL en Step Functions

Admite las ACL	No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Step Functions

Admite ABAC (etiquetas en las políticas)	Parcial

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte ¿Qué es ABAC? en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Utilizar credenciales temporales con Step Functions

Compatible con el uso de credenciales temporales	Sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluidas las que Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para más información sobre el cambio de roles, consulte Cambio a un rol (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para más información, consulte Credenciales de seguridad temporales en IAM.

Permisos principales entre servicios para Step Functions

Admite Forward access sessions (FAS)	Sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en él AWS, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones

con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información detallada sobre las políticas a la hora de realizar solicitudes de FAS, consulte Forward access sessions.

Roles de servicio para Step Functions

Compatible con roles de servicio	Sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Si se cambian los permisos de un rol de servicio, es posible que se interrumpa Step Functions la funcionalidad. Edite las funciones de servicio solo cuando se Step Functions proporcionen instrucciones para hacerlo.

Funciones vinculadas al servicio para Step Functions

Compatible con roles vinculados al servicio	No

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

¿Cómo AWS Step Functions funciona con IAM

Antes de utilizar IAM para gestionar el acceso Step Functions, infórmese sobre las funciones de IAM disponibles para su uso. Step Functions

Funciones de IAM que puede utilizar con AWS Step Functions

Característica de IAM	Step Functions soporte
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACL	No
ABAC (etiquetas en políticas)	Parcial
Credenciales temporales	Sí
Permisos de entidades principales	Sí
Roles de servicio	Sí
Roles vinculados al servicio	No

Para obtener una visión general de cómo Step Functions funcionan otros AWS servicios con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en la identidad para AWS Step Functions

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de Step Functions . Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS la API. Un administrador de IAM puede crear políticas

de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por cada uno de los tipos de recursos Step Functions, incluido el formato de los ARN para cada uno de los tipos de recursos, consulte las claves de condición, recursos y acciones de la Referencia de autorización de servicios. AWS Step Functions

Temas

- Prácticas recomendadas sobre las políticas
- Mediante la consola de Step Functions
- Cómo permitir a los usuarios consultar sus propios permisos

Prácticas recomendadas sobre las políticas

Las políticas basadas en la identidad determinan si alguien puede crear Step Functions recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.

- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utilice el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Política de validación de Analizador de acceso de IAM en la Guía de usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para más información, consulte Configuración del acceso a una API protegido por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Mediante la consola de Step Functions

Para acceder a la AWS Step Functions consola, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los Step Functions recursos de su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realicen llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para garantizar que los usuarios y los roles puedan seguir utilizando la Step Functions consola, adjunte también la política `ReadOnly` AWS gestionada Step Functions `ConsoleAccess` o la política

gestionada a las entidades. Para más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

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

Políticas basadas en la identidad para Step Functions

Compatibilidad con las políticas basadas en identidad	Sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está adjunto. Para más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en la identidad para Step Functions

Para ver ejemplos de políticas Step Functions basadas en la identidad, consulte. Ejemplos de políticas basadas en la identidad para AWS Step Functions

Políticas basadas en recursos incluidas Step Functions

Compatibilidad con las políticas basadas en recursos	No

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para obtener más información, consulte el tema Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

AWS políticas gestionadas para AWS Step Functions

Una política AWS administrada es una política independiente creada y administrada por AWS. AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

AWS política gestionada: AWSStepFunctionsConsoleFullAccess

Puede adjuntar la política `AWSStepFunctionsConsoleFullAccess` a las identidades de IAM.

Esta política otorga permisos de *administrador* que permiten a un usuario acceder para usar la consola Step Functions. Para disfrutar de una experiencia de consola completa, es posible que el usuario también necesite el PassRole permiso iam: para desempeñar otras funciones de IAM que pueda asumir el servicio.

AWS política gestionada: AWSStepFunctionsReadOnlyAccess

Puede adjuntar la política `AWSStepFunctionsReadOnlyAccess` a las identidades de IAM.

Esta política otorga permisos de *solo lectura* que permiten a un usuario o rol enumerar y describir las máquinas de estado, las actividades, las ejecuciones, las actividades, las etiquetas y los alias y las versiones de las máquinas de estado. MapRuns Esta política también otorga permiso para comprobar la sintaxis de las definiciones de máquinas de estado que proporciones.

AWS política gestionada: AWSStepFunctionsFullAccess

Puede adjuntar la política `AWSStepFunctionsFullAccess` a las identidades de IAM.

Esta política otorga permisos *completos* a un usuario o rol para usar la API Step Functions. Para un acceso completo, el usuario debe tener el PassRole permiso *iam:* en al menos un rol de IAM que pueda asumir el servicio.

Step Functions actualizaciones de las políticas gestionadas AWS

Consulte los detalles sobre las actualizaciones de las políticas AWS administradas Step Functions desde que este servicio comenzó a rastrear estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página de Step Functions Historial de documentos.

Cambio	Descripción	Fecha
AWSStepFunctionsReadOnlyAccess: actualización de una política actual	Step Functions se han añadido nuevos permisos que permiten llamar a la `states:ValidateStateMachineDefinition` API a una acción para	25 de abril de 2024

Cambio	Descripción	Fecha
	comprobar la sintaxis de las definiciones de máquinas de estados que proporciones.	
AWSStepFunctionsReadOnlyAccess: actualización de una política actual	Step Functions se han añadido nuevos permisos que permiten enumerar y leer datos relacionados con: etiquetas (ListTagsForResource), mapas distribuidos (ListMapseries, DescribeMapRun), versiones y alias (DescribeStateMachineAlias, ListStateMachineAliases, ListStateMachineVersions).	2 de abril de 2024
Step Functions comenzó a rastrear los cambios	Step Functions comenzó a realizar un seguimiento de los cambios de sus políticas AWS gestionadas.	2 de abril de 2024

Creación de un rol de IAM para la máquina de estado

AWS Step Functions puede ejecutar código y acceder a AWS los recursos (por ejemplo, invocar una AWS Lambda función). Para mantener la seguridad, debe conceder a Step Functions acceso a esos recursos mediante un rol de IAM.

Las funciones Tutoriales de Step Functions de IAM de esta guía le permiten aprovechar las funciones de IAM generadas automáticamente y que son válidas para la AWS región en la que creó la máquina de estados. No obstante, puede crear su propio rol de IAM para una máquina de estado.

Al crear una política de IAM para que la utilicen las máquinas de estado, la política debe incluir los permisos que desea que asuman las máquinas de estado. Puede utilizar una política AWS gestionada existente como ejemplo o puede crear una política personalizada desde cero que se adapte a sus necesidades específicas. Para obtener más información, consulte Creación de políticas de IAM en la Guía del usuario de IAM

Para crear su propio rol de IAM para una máquina de estado, siga los pasos que se describen en esta sección.

En este ejemplo, se crea un rol de IAM con permiso para invocar una función de Lambda.

Crear un rol para Step Functions

1. Inicie sesión en la consola de IAM y, a continuación, elija Roles, Crear rol.
2. En la página Seleccionar entidad de confianza, en Servicio de AWS , seleccione Step Functions en la lista y haga clic en Siguiente: Permisos.
3. En la página Attached permissions policy, elija Next: Review.
4. En la página Revisar, escriba `StepFunctionsLambdaRole` en Nombre del rol y, a continuación, elija Crear rol.

   El rol de IAM se muestra en la lista de roles.

Para obtener más información sobre los permisos y las políticas de IAM, consulte Administración de accesos en la Guía del usuario de IAM.

Prevención del problema del suplente confuso entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede provocar el confuso problema de un diputado. La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). Este tipo de suplantación puede ocurrir entre cuentas y entre servicios. El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder.

Para evitar que los agentes confusos, AWS proporciona herramientas que le ayudan a proteger sus datos para todos los servicios cuyos directores de servicio tienen acceso a los recursos de su cuenta. Esta sección se centra en la prevención específica de los problemas de los diputados confusos entre servicios AWS Step Functions; sin embargo, puede obtener más información sobre este tema en la sección de problemas de los diputados confusos de la Guía del usuario de la IAM.

Se recomienda utilizar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos para limitar los permisos que Step Functions

concede a otro servicio para el acceso a sus recursos. Utilice `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utilice `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:states:*:111122223333:*`.

Este es un ejemplo de una política de confianza que muestra cómo puede usar `aws:SourceArn` y `aws:SourceAccount` con Step Functions para evitar el problema del suplente confuso.

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

Asociación de una política insertada

Step Functions puede controlar otros servicios directamente en un estado de `Task`. Asocie políticas insertadas para permitir el acceso de Step Functions a las acciones de la API de los servicios que necesite controlar.

1. Abra la consola de IAM, elija Roles, busque el rol de Step Functions y selecciónelo.
2. Seleccione Añadir política insertada.
3. Utilice el Editor visual o la pestaña JSON para crear políticas para el rol.

Para obtener más información sobre cómo AWS Step Functions puede controlar otros AWS servicios, consulte. Uso AWS Step Functions con otros servicios

> **Note**
>
> Para ver ejemplos de políticas de IAM creadas por la consola de Step Functions, consulte Políticas de IAM para servicios integrados.

Creación de permisos granulares de IAM para usuarios no administradores

Las políticas gestionadas por defecto en IAM, por ejemplo`ReadOnly`, no cubren todos los tipos de AWS Step Functions permisos. En esta sección se describen los distintos tipos de permisos y se facilitan algunas configuraciones de ejemplo.

Step Functions tiene cuatro categorías de permisos. En función del acceso que desee proporcionar a un usuario, puede controlar el acceso mediante la utilización de permisos en estas categorías.

Permisos de nivel de servicio

Se aplican a los componentes de la API que no actúan en un recurso específico.

Permisos de nivel de máquina de estado

Se aplican a todos los componentes de la API que actúan sobre una máquina de estado específica.

Permisos de nivel de ejecución

Se aplican a todos los componentes de la API que actúan sobre una ejecución específica.

Permisos de nivel de actividad

Se aplican a todos los componentes de la API que actúan sobre una actividad específica o sobre una instancia concreta de una actividad.

Permisos de nivel de servicio

Este nivel de permiso se aplica a todas las acciones de la API que no actúan en un recurso específico. Estas incluyen `CreateStateMachineCreateActivity`, `ListStateMachinesListActivities`, y`ValidationStateMachineDefinition`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:ListActivities",
        "states:CreateStateMachine",
        "states:CreateActivity",
        "states:ValidationStateMachineDefinition",
      ],
      "Resource": [
        "arn:aws:states:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam:::role/my-execution-role"
      ]
    }
  ]
}
```

Permisos de nivel de máquina de estado

Este nivel de permiso se aplica a todas las acciones de la API que actúan sobre una máquina de estado específica. Estas operaciones de la API requieren el Nombre de recurso de Amazon (ARN) de la máquina de estado como parte de la solicitud, como por ejemplo `DeleteStateMachine`, `DescribeStateMachine`, `StartExecution` y `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*"
      ]
    }
  ]
}
```

Permisos de nivel de ejecución

Este nivel de permiso se aplica a todas las acciones de la API que actúan sobre una ejecución específica. Estas operaciones del API requieren el ARN de la ejecución como parte de la solicitud, por ejemplo `DescribeExecution`, `GetExecutionHistory` y `StopExecution`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
```

```
}
```

Permisos de nivel de actividad

Este nivel de permiso se aplica a todas las acciones de la API que actúan sobre una actividad específica o en una instancia particular de la misma. Estas operaciones de la API requieren el ARN de la actividad o el token de la instancia como parte de la solicitud, como por ejemplo `DeleteActivity`, `DescribeActivity`, `GetActivityTask` y `SendTaskHeartbeat`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
      ]
    }
  ]
}
```

Acceder a los recursos de otras partes Cuentas de AWS de sus flujos de trabajo

Step Functions proporciona acceso multicuenta a los recursos configurados Cuentas de AWS en diferentes flujos de trabajo. Con las integraciones de servicios de Step Functions, puedes invocar cualquier AWS recurso multicuenta, incluso si Servicio de AWS no es compatible con políticas basadas en recursos o llamadas entre cuentas.

Por ejemplo, supongamos que tiene dos, denominados Desarrollo y Cuentas de AWS Pruebas, al mismo tiempo. Región de AWS Al utilizar el acceso entre cuentas, el flujo de trabajo de la cuenta Desarrollo puede acceder a recursos, como los buckets de Amazon S3, las tablas de Amazon DynamoDB y las funciones de Lambda, que estén disponibles en la cuenta Pruebas.

> **Important**
>
> Los roles de IAM y las políticas basadas en recursos delegan el acceso entre cuentas solo dentro de una única partición. Suponga, por ejemplo, que tiene una cuenta en EE. UU. Oeste (Norte de California) en la partición estándar `aws`. También tiene una cuenta en China (Pekín) en la partición `aws-cn`. No puede utilizar una política de Amazon S3 basada en recursos en su cuenta en China (Pekín) para permitir el acceso a los usuarios de su cuenta `aws` estándar.

Para obtener más información sobre el acceso entre cuentas, consulte Lógica de evaluación de políticas entre cuentas en la Guía del usuario de IAM.

Aunque cada una Cuenta de AWS mantiene el control total sobre sus propios recursos, con Step Functions, puede reorganizar, intercambiar, añadir o eliminar pasos de sus flujos de trabajo sin necesidad de personalizar ningún código. Puede hacerlo incluso cuando cambien los procesos o evolucionen las aplicaciones.

También puede invocar ejecuciones de máquinas de estado anidadas para que estén disponibles entre diferentes cuentas. Al hacerlo así, separa y aísla los flujos de trabajo de manera eficiente. Cuando se utiliza el patrón de integración de servicios `.sync` en los flujos de trabajo que acceden a otro flujo de trabajo de Step Functions en una cuenta diferente, Step Functions utiliza un sondeo que consume la cuota asignada. Para obtener más información, consulte Ejecutar un trabajo (.sync).

> **Note**
>
> Actualmente, la integración AWS del SDK entre regiones y el acceso a los AWS recursos entre regiones no están disponibles en Step Functions.

Contenido

Conceptos clave de este tema

Rol de ejecución

Un rol de IAM que Step Functions utiliza para ejecutar código y acceder a AWS recursos, como la acción Invoke de la AWS Lambda función.

Integración con los servicios

Las acciones de la API de integración del AWS SDK a las que se puede llamar desde un `Task` estado de tus flujos de trabajo.

cuenta de origen

Una Cuenta de AWS que es propietaria de la máquina de estados y que ha comenzado su ejecución.

cuenta de destino

Y Cuenta de AWS a la que se hacen llamadas entre cuentas.

rol de destino

Un rol de IAM en la cuenta de destino que la máquina de estado asume para realizar llamadas a recursos de los que es propietaria la cuenta de destino.

Ejecutar un trabajo (.sync)

Un patrón de integración de servicios que se utiliza para llamar a servicios, como AWS Batch. También hace que una máquina de estado de Step Functions espere a que se complete un trabajo antes de pasar al siguiente estado. Para indicar que Step Functions debe esperar, añada el sufijo `.sync` al campo `Resource` de la definición de estado `Task`.

Invocación de recursos entre cuentas

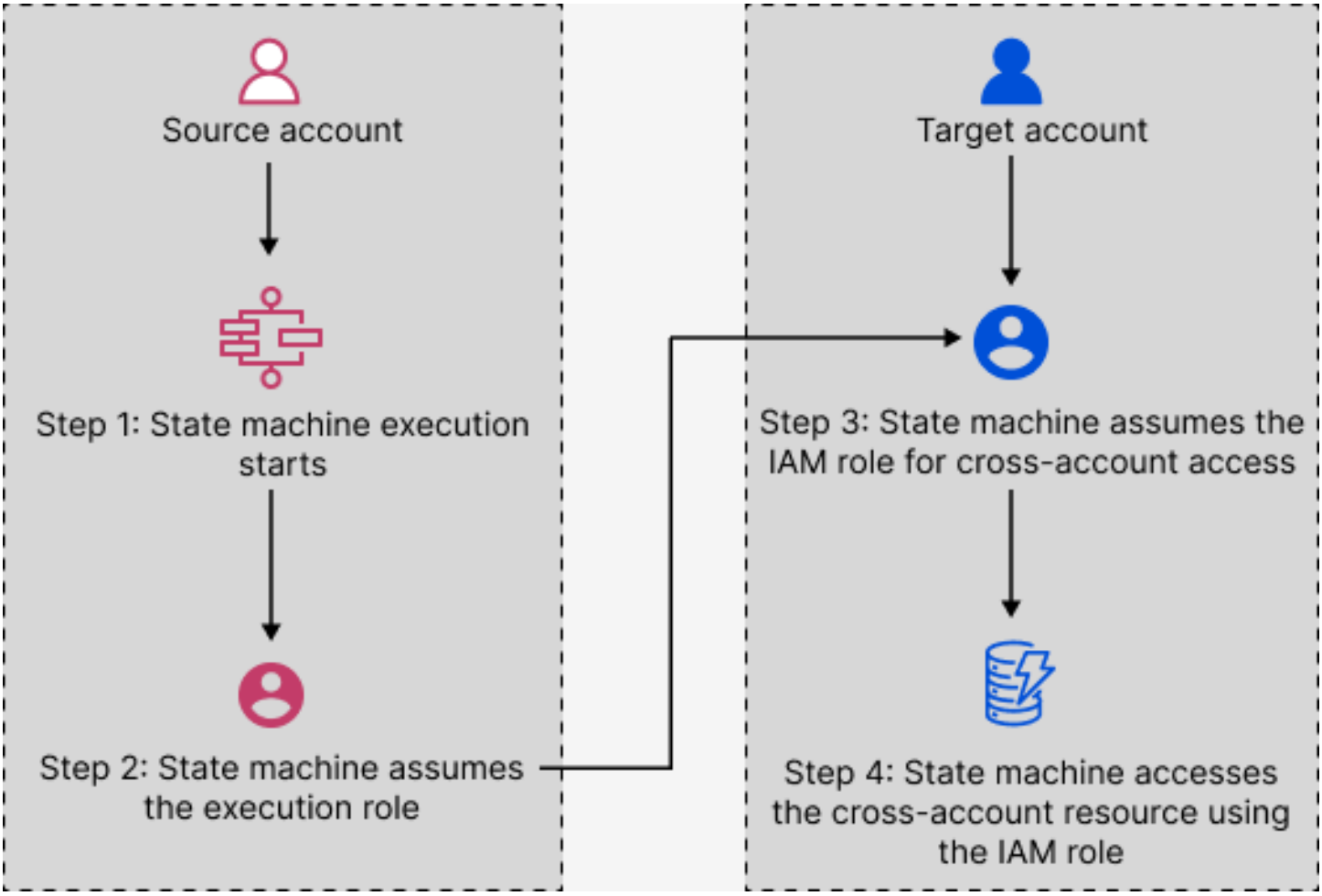
Para invocar un recurso entre cuentas en los flujos de trabajo, haga lo siguiente:

1. Cree un rol de IAM en la cuenta de destino que contiene el recurso. Esta función otorga a la cuenta de origen, que contiene la máquina de estado, permisos para acceder a los recursos de la cuenta de destino.
2. En la definición del estado `Task`, especifique el rol de IAM de destino que asumirá la máquina de estado antes de invocar el recurso entre cuentas.

3. Modifique la política de confianza del rol de IAM de destino para permitir que la cuenta de origen asuma esta función temporalmente. La política de confianza debe incluir el nombre de recurso de Amazon (ARN) de la máquina de estado definida en la cuenta de origen. Además, defina los permisos adecuados en la función de IAM de destino para llamar al AWS recurso.
4. Actualice el rol de ejecución de la cuenta de origen para incluir el permiso necesario para asumir el rol de IAM de destino.

Para ver un ejemplo, consulte Tutorial: Acceso a recursos multicuenta AWS.

> **Note**
>
> Puede configurar la máquina de estado para que asuma un rol de IAM para acceder a los recursos desde varias Cuentas de AWS. No obstante, una máquina de estado solo puede asumir un rol de IAM en un momento dado.



Tutorial: Acceso a recursos multicuenta AWS

Con la compatibilidad con acceso entre cuentas en Step Functions, puede compartir recursos configurados en diferentes Cuentas de AWS. En este tutorial, explicamos el proceso de acceso a una función de Lambda entre cuentas definida en una cuenta denominada Producción. Esta función se invoca desde una máquina de estado en una cuenta llamada Desarrollo. En este tutorial, se hace referencia a la cuenta Desarrollo como cuenta de origen y a la cuenta Producción como cuenta de destino que contiene el rol de IAM de destino.

Para empezar, en la definición del estado `Task` debe especificar el rol de IAM de destino que debe asumir la máquina de estado antes de invocar la función de Lambda entre cuentas. A continuación, modifique la política de confianza en el rol de IAM de destino para permitir que la cuenta de origen asuma el rol de destino temporalmente. Además, para llamar al AWS recurso, defina los permisos adecuados en la función de IAM de destino. Por último, actualice el rol de ejecución de la cuenta de origen para especificar el permiso necesario para asumir el rol de destino.

Puede configurar la máquina de estado para que asuma un rol de IAM para acceder a los recursos desde varias Cuentas de AWS. No obstante, una máquina de estado solo puede asumir un rol de IAM en un momento dado sobre la base de la definición de estado `Task`.

> **Note**
>
> Actualmente, la integración AWS del SDK entre regiones y el acceso a los AWS recursos entre regiones no están disponibles en Step Functions.

Contenido

Requisitos previos

- En este tutorial se utiliza el ejemplo de una función de Lambda para demostrar cómo configurar el acceso entre cuentas. Puedes usar cualquier otro AWS recurso, pero asegúrate de haber configurado el recurso en una cuenta diferente.

  > **Important**
  >
  > Los roles de IAM y las políticas basadas en recursos delegan el acceso entre cuentas solo dentro de una única partición. Suponga, por ejemplo, que tiene una cuenta en EE. UU. Oeste (Norte de California) en la partición estándar `aws`. También tiene una cuenta en China (Pekín) en la partición `aws-cn`. No puede utilizar una política de Amazon S3 basada en recursos en su cuenta en China (Pekín) para permitir el acceso a los usuarios de su cuenta `aws` estándar.

- Anote el Nombre de recurso de Amazon (ARN) del recurso entre cuentas en un archivo de texto. Más adelante en este tutorial, proporcionará este ARN en la definición de estado `Task` de la máquina de estado. El siguiente es un ejemplo de un ARN de función de Lambda:

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

- Asegúrese de haber creado el rol de IAM de destino que debe asumir la máquina de estado.

Paso 1: actualice la definición del estado de la tarea para especificar el rol de destino

En el estado `Task` del flujo de trabajo, añada un campo `Credentials` que contenga la identidad que debe asumir la máquina de estado antes de invocar la función de Lambda entre cuentas.

El siguiente procedimiento demuestra cómo acceder a una función de Lambda entre cuentas llamada `Echo`. Puedes llamar a cualquier AWS recurso siguiendo estos pasos.

1. Abra la consola de Step Functions y seleccione Crear máquina de estado.
2. En la página Elegir un método de creación, elija Diseñe su flujo de trabajo de forma visual y mantenga todas las selecciones predeterminadas.
3. Para abrir Workflow Studio, elija Siguiente.
4. En la pestaña Acciones, arrastre y suelte un estado `Task` en el lienzo. Esto invoca la función de Lambda entre cuentas que utiliza este estado `Task`.
5. En la pestaña Configuración, haga lo siguiente:

a. Cambie el nombre del estado a **`Cross-account call`**.

b. En Nombre de la función, elija Introducir nombre de la función y, a continuación, introduzca el ARN de la función de Lambda en el cuadro. Por ejemplo, `arn:aws:lambda:us-east-2:111122223333:function:Echo`.

c. En Proporcionar ARN del rol de IAM, especifique el ARN del rol de IAM de destino. Por ejemplo, `arn:aws:iam::111122223333:role/LambdaRole`.

> **Tip**
>
> También puede especificar una ruta de referencia a un par clave-valor existente en la entrada JSON del estado que contiene el ARN del rol de IAM. Para ello, elija Obtener ARN del rol de IAM en tiempo de ejecución a partir de la entrada de estado. Para ver un ejemplo de cómo especificar un valor mediante una ruta de referencia, consulte Especificar JSONPath como un ARN de rol de IAM.

6. Elija Siguiente.
7. En la página Revisar código generado, elija Siguiente.
8. En la página Especificar configuración de la máquina de estado, especifique los detalles de la nueva máquina de estado, como el nombre, los permisos y el nivel de registro.
9. Elija Crear máquina de estado.
10. Anote el ARN del rol de IAM de la máquina de estado y el ARN de la máquina de estado en un archivo de texto. Deberá incluir estos ARN en la política de confianza de la cuenta de destino.

La definición de estado `Task` debería ser similar a la definición siguiente.

```
{
  "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
    }
  }
}
```

Paso 2: actualice la política de confianza del rol de destino

El rol de IAM debe existir en la cuenta de destino; debe modificar la política de confianza para permitir que la cuenta de origen asuma este rol temporalmente. Además, puede controlar quién puede asumir el rol de IAM de destino.

Tras crear la relación de confianza, un usuario de la cuenta de origen puede utilizar la operación de la AssumeRoleAPI AWS Security Token Service (AWS STS). Esta operación proporciona credenciales de seguridad temporales que permiten el acceso a AWS los recursos de una cuenta de destino.

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la consola, elija Funciones y, a continuación, utilice el cuadro de búsqueda para buscar el rol de IAM de destino. Por ejemplo, *`LambdaRole`*.
3. Seleccione la pestaña Relaciones de confianza.
4. Elija Editar política de confianza y pegue la siguiente política de confianza. Asegúrese de sustituir el Cuenta de AWS número y el ARN del rol de IAM. El campo `sts:ExternalId` controla, además, quién puede asumir el rol. El nombre de la máquina de estados debe incluir solo los caracteres compatibles con la AWS Security Token Service `AssumeRole` API. Para obtener más información, consulta AssumeRolela referencia de la AWS Security Token Service API.

   ```
   {
     "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. Mantenga esta ventana abierta y continúe con el siguiente paso para realizar más acciones.

Paso 3: agregue el permiso necesario en el rol de destino

Los permisos de las políticas de IAM determinan si una solicitud específica se permite o se deniega. El rol de IAM de destino debe tener el permiso correcto para invocar la función de Lambda.

1. Elija la pestaña Permisos.
2. Seleccione Agregar permisos y, a continuación, Crear política insertada.
3. Elija la pestaña JSON y reemplace el contenido existente por el permiso siguiente. No olvide de reemplazar el ARN de la función de Lambda.

```
{
  "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. Elija Revisar política.
5. En la página Revisar política, escriba un nombre y luego elija Crear política.

Paso 4: agregue un permiso en el rol de ejecución para asumir el rol de destino

Step Functions no genera automáticamente la AssumeRolepolítica para todas las integraciones de servicios multicuenta. Debe añadir el permiso necesario al rol de ejecución de la máquina de estado para que pueda asumir un rol de IAM de destino en una o más Cuentas de AWS.

1. Abra el rol de ejecución de la máquina de estado en la consola de IAM en https://console.aws.amazon.com/iam/. Para ello:
   a. Abra la máquina de estado que creó en el Paso 1 en la cuenta de origen.
   b. En la página Detalle de la máquina de estado, elija ARN del rol de IAM.
2. En la pestaña Permisos, elija Agregar permisos y luego Crear política insertada.
3. Elija la pestaña JSON y reemplace el contenido existente por el permiso siguiente. No olvide de reemplazar el ARN de la función de Lambda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/LambdaRole"  // The target role
 to be assumed
    }
  ]
}
```

4. Elija Revisar política.
5. En la página Revisar política, escriba un nombre y luego elija Crear política.

Acceso entre cuentas para el patrón de integración .sync

Cuando se utilizan los patrones de integración de servicios `.sync` en los flujos de trabajo, Step Functions sondea el recurso entre cuentas invocado para confirmar que la tarea se ha completado. Esto provoca un ligero retraso entre el momento real de finalización de la tarea y el momento en que Step Functions reconoce la tarea como completada. El rol de IAM de destino necesita los permisos necesarios para que una invocación de `.sync` complete este bucle de sondeo. Para ello, el rol de IAM de destino debe tener una política de confianza que permita que la cuenta de origen lo asuma.

Además, el rol de IAM de destino necesita los permisos requeridos para completar el bucle de sondeo.

> **Note**
>
> Para flujos de trabajo rápidos anidados, `arn:aws:states:::states:startExecution.sync` no es compatible actualmente. En su lugar, use `arn:aws:states:::aws-sdk:sfn:startSyncExecution`.

Actualización de la política de confianza para llamadas .sync

Actualice la política de confianza del rol de IAM de destino como se muestra en el siguiente ejemplo. El campo `sts:ExternalId` controla, además, quién puede asumir el rol. El nombre de la máquina de estados debe incluir solo los caracteres compatibles con la AWS Security Token Service `AssumeRole` API. Para obtener más información, consulta AssumeRolela referencia de la AWS Security Token Service API.

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

Se requieren permisos para las llamadas .sync

Para conceder los permisos necesarios para la máquina de estado, actualice los permisos necesarios para el rol de IAM de destino. Para obtener más información, consulte the section called

"Políticas de IAM para servicios integrados". No se requieren EventBridge los permisos de Amazon de las políticas de ejemplo. Por ejemplo, para iniciar una máquina de estado, añada los siguientes permisos.

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

Puntos de conexión de VPC para Step Functions

Si utiliza Amazon Virtual Private Cloud (Amazon VPC) para alojar sus AWS recursos, puede establecer una conexión entre su Amazon VPC y los flujos de trabajo. AWS Step Functions Puede utilizar esta conexión con sus flujos de trabajo de Step Functions sin cruzar la Internet pública. Los puntos de conexión de VPC son compatibles con los flujos de trabajo estándar, los flujos de trabajo rápidos y los flujos de trabajo rápidos sincrónicos.

Amazon VPC le permite lanzar AWS recursos en una red virtual personalizada. Puede utilizar una VPC para controlar la configuración de red, como el intervalo de direcciones IP, las subredes, las tablas de enrutamiento y las puertas de enlace de red. Para obtener más información sobre VPC, consulte la Guía del usuario de Amazon VPC.

Para conectar su Amazon VPC a Step Functions, primero debe definir un punto de enlace de VPC de interfaz, que le permita conectar su VPC a otros servicios. AWS El punto de conexión ofrece conectividad escalable de confianza sin necesidad de utilizar una gateway de Internet, una instancia de conversión de las direcciones de red (NAT) o una conexión de VPN. Para obtener más información, consulte Puntos de conexión de VPC de la interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Creación del punto de conexión

Puede crear un AWS Step Functions punto de conexión en su VPC mediante AWS Management Console, el AWS Command Line Interface (AWS CLI), un AWS SDK, la AWS Step Functions API o. AWS CloudFormation

Para obtener información sobre la creación y configuración de un punto de conexión mediante la consola de Amazon VPC o la AWS CLI, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

> **Note**
>
> Al crear el punto de conexión, especifique que Step Functions es el servicio al que desea que se conecte la VPC. En la consola de Amazon VPC, los nombres de los servicios varían en función de la AWS región. Por ejemplo, si elige Este de EE. UU. (Norte de Virginia), el nombre de servicio para flujos de trabajo estándar y flujos de trabajo rápidos es com.amazonaws.us-east-1.states y el nombre de servicio para flujos de trabajo rápido síncronos es com.amazonaws.us-east-1.sync-states.

> **Note**
>
> Es posible usar puntos de conexión de VPC sin anular el punto de conexión en el SDK a través DNS privado. No obstante, si desea anular el punto de conexión en el SDK para flujos de trabajo rápidos síncronos, debe establecer la configuración `DisableHostPrefixInjection` en `true`. Ejemplo (Java SDK 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();
```

Para obtener información sobre cómo crear y configurar un punto final mediante AWS CloudFormation, consulte el recurso AWS: :EC2: :VPCendpoint en la Guía del usuario.AWS CloudFormation

Políticas de punto de conexión de VPC de Amazon

Para controlar el acceso de conectividad a Step Functions, puede adjuntar una política de punto final AWS Identity and Access Management (IAM) mientras crea un punto final de Amazon VPC. Puede crear reglas de IAM complejas al asociar varias políticas de punto de conexión. Para obtener más información, consulte:

- Políticas de punto de conexión de Amazon Virtual Private Cloud para Step Functions
- Creación de permisos granulares de IAM para usuarios no administradores
- Control del acceso a los servicios con puntos de conexión de VPC

Políticas de punto de conexión de Amazon Virtual Private Cloud para Step Functions

Puede crear una política de punto de conexión de Amazon VPC para Step Functions en la que especifique lo siguiente:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- El recurso en el que se pueden realizar las acciones.

En el ejemplo siguiente se muestra una política de punto de conexión de VPC Amazon que permite a un usuario crear máquinas de estado y deniega a todos los demás usuarios el permiso para eliminar máquinas de estado. La política de ejemplo también concede permiso de ejecución a todos los usuarios de .

```
{
```

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

Para obtener más información acerca de la creación de políticas de punto de enlace, consulte lo siguiente:

- Creación de permisos granulares de IAM para usuarios no administradores
- Control del acceso a los servicios con puntos de conexión de VPC

Políticas de IAM para servicios integrados

Al crear una máquina de estados en la AWS Step Functions consola, Step Functions genera una política AWS Identity and Access Management (IAM) basada en los recursos utilizados en la definición de la máquina de estados, de la siguiente manera:

- Si su máquina de estados usa una de las integraciones optimizadas, Step Functions creará una política con los permisos y roles necesarios para su máquina de estados. (Excepción: MediaConvert la integración requiere que configure los permisos manualmente; consultePolíticas de IAM para AWS Elemental MediaConvert)

- Si su máquina de estado usa una de las integraciones del AWS SDK, se creará un rol de IAM con permisos parciales. Después, puede utilizar la consola de IAM para añadir cualquier política de rol que falte.

En los siguientes ejemplos se muestra cómo Step Functions genera una política de IAM en función de la definición de la máquina de estado. Los elementos del código de ejemplo como *`[[resourceName]]`* se sustituyen por los recursos estáticos incluidos en la definición de la máquina de estado. Si tiene varios recursos estáticos, habrá una entrada para cada uno en el rol de IAM.

Recursos dinámicos frente a recursos estáticos

Los recursos estáticos se definen directamente en la estado de la tarea de la máquina de estado. Cuando se incluye la información sobre los recursos a los que se llama directamente en los estados de la tarea, Step Functions crea un rol de IAM exclusivo para esos recursos.

Los recursos dinámicos son los que se transfieren a la entrada del estado y se puede acceder a ellos mediante una ruta (consulte Rutas). Si pasa recursos dinámicos a una tarea, Step Functions creará una política más privilegiada que especifica: `"Resource": "*"`.

Permisos adicionales para las tareas que utilizan el patrón de ejecución de un trabajo

Para las tareas que utilizan el patrón de ejecución de un trabajo (las que terminan en `.sync`), se necesitan permisos adicionales para monitorizar y recibir una respuesta de las acciones de la API de los servicios conectados. Las políticas relacionadas incluyen más permisos que las tareas que utilizan los patrones de solicitud de respuesta o espera de devolución de llamada. Consulte Patrones de integración de servicios para obtener información acerca de las tareas síncronas.

> **Note**
>
> Debe proporcionar permisos adicionales para las integraciones de servicios que admitan el patrón Run a Job (.sync).

Step Functions utiliza dos métodos para supervisar el estado de un trabajo cuando se ejecuta en un servicio conectado: sondeos y eventos.

El sondeo requiere permiso para las acciones de la API `Describe` o `Get`, como `ecs:DescribeTasks` o `glue:GetJobRun`. Si el rol no tiene estos permisos, es posible que Step

Functions no pueda determinar el estado del trabajo. Esto se debe a que algunas integraciones del servicio Run a Job (.sync) no admiten EventBridge eventos y algunos servicios solo envían eventos según el mejor esfuerzo.

Los eventos enviados desde AWS los servicios a Amazon EventBridge se dirigen a Step Functions mediante una regla gestionada y requieren permisos para `events:PutTargetsevents:PutRule`, y`events:DescribeRule`. Si no tiene estos permisos en el rol, es posible que se produzca un retraso antes de que Step Functions tenga conocimiento de que ha completado el trabajo. Para obtener más información sobre EventBridge los eventos, consulte Eventos de AWS los servicios.

> **Note**
>
> En el caso de las tareas de ejecución de un trabajo (.sync) que admiten sondeos y eventos, es posible que la tarea se complete correctamente mediante eventos. Esto puede ocurrir incluso si el rol carece de los permisos necesarios para realizar sondeos En este caso, es posible que no se dé cuenta inmediatamente de que los permisos de sondeos faltan o son incorrectos. En la instancia infrecuente de que Step Functions no entregue o procese el evento, la ejecución podría bloquearse. Para comprobar que sus permisos de sondeo están configurados correctamente, puede ejecutar una ejecución en un entorno sin EventBridge eventos de las siguientes maneras:
>
> - Elimine la regla gestionada de EventBridge, que es responsable de reenviar los eventos a Step Functions. Todas las máquinas de estado de la cuenta comparten esta regla gestionada, por lo que debe realizar esta acción únicamente en una cuenta de prueba o de desarrollo para evitar cualquier impacto involuntario en otras máquinas de estado. Puede identificar la regla administrada específica que desea eliminar inspeccionando el campo `Resource` utilizado para `events:PutRule` en la plantilla de política del servicio de destino. La regla administrada se volverá a crear la próxima vez que cree o actualice una máquina de estado que utilice esa integración de servicios. Para obtener más información sobre la eliminación de EventBridge reglas, consulte Desactivar o eliminar una regla.
> - Realice pruebas con Step Functions Local, que no admite el uso de eventos para completar las tareas de ejecución de trabajo (.sync). Para usar Step Functions Local, asuma el rol de IAM que utilice la máquina de estado. Es posible que deba editar la Relación de confianza. Defina las variables de entorno `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `AWS_SESSION_TOKEN` en los valores del rol asumido y, a continuación, inicie Step Functions Local usando java -jar StepFunctionsLocal.jar. Por último, utilice el --endpoint-url parámetro AWS CLI with para crear una máquina de

estados, iniciar una ejecución y obtener el historial de ejecuciones. Para obtener más información, consulte Probar máquinas de estado de forma local.

Si se detiene una tarea que utiliza el patrón de ejecución de trabajo (.sync), Step Functions hará todo lo posible por cancelar la tarea. Esto requiere permiso para las acciones de la API `Cancel`, `Stop`, `Terminate` o `Delete`, como `batch:TerminateJob` o `eks:DeleteCluster`. Si la función no tiene estos permisos, Step Functions no podrá cancelar la tarea y podrían acumularse cargos adicionales mientras siga ejecutándose. Para obtener más información, consulte Ejecución de un trabajo.

Plantillas de políticas utilizadas para crear roles de IAM

En los temas siguientes se incluyen las plantillas de políticas que se utilizarán cuando decida que Step Functions cree un nuevo rol por usted.

> **Note**
>
> Revisa estas plantillas para entender cómo Step Functions crea tus políticas de IAM y como ejemplo de cómo crear manualmente políticas de IAM para Step Functions cuando trabajas con otros AWS servicios. Para obtener más información sobre las integraciones de servicios de Step Functions, consulte Uso AWS Step Functions con otros servicios.

Temas

- Políticas de IAM para Amazon API Gateway
- Políticas de IAM para Amazon Athena
- Políticas de IAM para AWS Batch
- IAM policies for Amazon Bedrock
- Políticas de IAM para AWS CodeBuild
- Políticas de IAM para Amazon DynamoDB
- Políticas de IAM para Amazon ECS/AWS Fargate
- Políticas de IAM para Amazon EKS
- Políticas de IAM para Amazon EMR
- Políticas de IAM para Amazon EMR en EKS
- Políticas de IAM para Amazon EMR Serverless

- Políticas de IAM para Amazon EventBridge
- Políticas de IAM para AWS Lambda
- Políticas de IAM para AWS Elemental MediaConvert
- Políticas de IAM para AWS Glue
- Políticas de IAM para AWS Glue DataBrew
- Políticas de IAM para Amazon SageMaker
- Políticas de IAM para Amazon SNS
- Políticas de IAM para Amazon SQS
- Políticas de IAM para AWS Step Functions
- Políticas de IAM para AWS X-Ray
- Actividades o sin tareas

Políticas de IAM para Amazon API Gateway

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Recursos:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

En el siguiente ejemplo de código se muestra una política de recursos para llamar a API Gateway.

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

Políticas de IAM para Amazon Athena

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

`StartQueryExecution`

Recursos estáticos

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": [
            "*"
        ]
    }
]
}
```

Recursos dinámicos

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`

Recursos

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
```

```
            "athena:stopQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryExecution

Recursos

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryResults

Recursos

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

Políticas de IAM para AWS Batch

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Debido a que AWS Batch proporciona soporte parcial para el control de acceso a nivel de recursos, debe usarlo. `"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

Al crear una máquina de estado mediante la consola, Step Functions crea automáticamente un rol de ejecución para la máquina de estado con los privilegios mínimos necesarios. Estas IAM funciones generadas automáticamente son válidas para la máquina Región de AWS en la que se crea la máquina de estados.

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Le recomendamos que, al crear políticas de IAM, no incluya caracteres comodín en las políticas. Como práctica recomendada de seguridad, debe reducir el alcance de las políticas en la medida de lo posible. Debe usar políticas dinámicas solo cuando no se conozcan ciertos parámetros de entrada durante el tiempo de ejecución.

En este tema

Ejemplos de políticas de IAM para la integración de Amazon Bedrock con Step Functions

En la siguiente sección se describen los permisos de IAM que necesita según la API de Amazon Bedrock que utilice para un modelo fundacional o aprovisionado específico. En esta sección también se incluyen ejemplos de políticas que otorgan acceso total.

Recuerde reemplazar el texto en *`cursiva`* por la información específica del recurso.

IAMejemplo de política para acceder a un modelo básico específico mediante InvokeModel

El siguiente es un ejemplo IAM de política para una máquina de estados que accede a un modelo básico específico denominado `amazon.titan-text-express-v1` mediante la acción de la InvokeModelAPI.

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

IAMejemplo de política para acceder a un modelo aprovisionado específico mediante InvokeModel

El siguiente es un ejemplo IAM de política para una máquina de estados que accede a un modelo aprovisionado específico denominado `c2oi931ulksx` mediante la acción de la InvokeModelAPI.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

Ejemplo de IAM política de acceso completo a utilizar InvokeModel

El siguiente es un ejemplo IAM de política para una máquina de estados que proporciona acceso total cuando se utiliza la acción de la InvokeModelAPI.

```
{
    "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/*"
            ]
        }
    ]
}
```

Ejemplo de política de IAM para acceder a un modelo fundacional específico como modelo base

El siguiente es un ejemplo de IAM política para que una máquina de estados acceda a un modelo básico específico `amazon.titan-text-express-v1` denominado modelo base mediante la acción de la CreateModelCustomizationJobAPI.

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

Ejemplo de política de IAM para acceder a un modelo personalizado específico como modelo base

El siguiente es un ejemplo IAM de política para que una máquina de estados acceda a un modelo personalizado específico como modelo base mediante la acción de la CreateModelCustomizationJobAPI.

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

Ejemplo de IAM política de acceso completo para usar CreateModelCustomizationJob .sync

El siguiente es un ejemplo IAM de política para una máquina de estados que proporciona acceso total cuando se utiliza la acción de la CreateModelCustomizationJobAPI.

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

IAMejemplo de política para acceder a un modelo básico específico mediante CreateModelCustomizationJob .sync

El siguiente es un ejemplo IAM de política para que una máquina de estados acceda a un modelo básico específico denominado `amazon.titan-text-express-v1` mediante la acción de la API CreateModelCustomizationJob.sync.

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

IAMejemplo de política para acceder a un modelo personalizado mediante .sync CreateModelCustomizationJob

El siguiente es un ejemplo IAM de política para que una máquina de estados acceda a un modelo personalizado mediante la acción de la API CreateModelCustomizationJob.sync.

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

Ejemplo de IAM política de acceso completo para usar .sync CreateModelCustomizationJob

El siguiente es un ejemplo IAM de política para una máquina de estados que proporciona acceso total cuando se utiliza la acción de la API CreateModelCustomizationJob.sync.

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

Políticas de IAM para AWS CodeBuild

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Recursos:

```
{
    "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`

Recursos estáticos

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]]"
      ]
    }
  ]
}
```

Recursos dinámicos

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`

Recursos estáticos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Recursos dinámicos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
```

```
}
```

`BatchDeleteBuilds`

Recursos estáticos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Recursos dinámicos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchGetReports`

Recursos estáticos

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:report-group/[[reportName]]"
      ]
    }
  ]
}
```

Recursos dinámicos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:report-group/*"
      ]
    }
  ]
}
```

`StartBuildBatch`

Recursos estáticos

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]]"
            ]
        }
    ]
}
```

Recursos dinámicos

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

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
```

```
}
```

`RetryBuildBatch`

Recursos estáticos

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]]"
            ]
        }
    ]
}
```

Recursos dinámicos

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`

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Políticas de IAM para Amazon DynamoDB

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de su definición de máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Recursos estáticos

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:[[region]]:[[accountId]]:table/[[tableName]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
}
```

Para obtener más información sobre las políticas de IAM para todas las acciones de la API de DynamoDB, consulte Políticas de IAM con DynamoDB en la Guía para desarrolladores de Amazon DynamoDB. Además, para obtener información sobre las políticas de IAM para PartiQL para DynamoDB, consulte Políticas de IAM con PartiQL para DynamoDB en la Guía para desarrolladores de Amazon DynamoDB.

Políticas de IAM para Amazon ECS/AWS Fargate

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Dado que el valor de `TaskId` no se conoce hasta que se envía la tarea, Step Functions crea una política de `"Resource": "*"` con más privilegios.

> **Note**
>
> Solo se puede detener tareas de Amazon Elastic Container Service (Amazon ECS) que se hayan iniciado con Step Functions, independientemente de la política de IAM `"*"`.

Run a Job (.sync)

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]:[[revisionNumber]]"
            ]
        },
        {
            "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"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "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)

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]:[[revisionNumber]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
}
```

Si sus tareas programadas de Amazon ECS requieren el uso de una función de ejecución de tareas, una función de tarea o una anulación de la función de tarea, debe añadir `iam:PassRole` permisos para cada función de ejecución de tareas, función de tarea o anulación de función de tarea a la

función IAM de CloudWatch eventos de la entidad que realiza la llamada, que en este caso es Step Functions.

Políticas de IAM para Amazon EKS

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

`CreateCluster`

Recursos

```
{
    "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`

Recursos

```
{
    "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`

Recursos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteCluster",
                "eks:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:cluster/ExampleCluster"
            ]
        }
    ]
}
```

`DeleteNodegroup`

Recursos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteNodegroup",
```

```
                "eks:DescribeNodegroup"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:nodegroup/ExampleCluster/
ExampleNodegroup/*"
            ]
        }
    ]
}
```

Para obtener más información sobre cómo usar Amazon EKS con Step Functions, consulte Llamar a Amazon EKS con Step Functions.

Políticas de IAM para Amazon EMR

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

`addStep`

Recursos estáticos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Recursos dinámicos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

`cancelStep`

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
```

```
        }
    ]
}
```

createCluster

Recursos estáticos

```
{
  "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

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
```

```
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceFleetByName`

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceGroupByName`

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
```

```
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

`terminateCluster`

Recursos estáticos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Recursos dinámicos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

Políticas de IAM para Amazon EMR en EKS

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina estatal. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

`CreateVirtualCluster`

Recursos

```
{
    "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`

Recursos estáticos

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]]"
      ]
    }
  ]
}
```

Recursos dinámicos

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`

Recursos estáticos

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]]"
          ]
        }
```

```
      }
    }
  ]
}
```

Recursos dinámicos

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

Políticas de IAM para Amazon EMR Serverless

Al crear una máquina de estado mediante la consola, Step Functions crea automáticamente un rol de ejecución para la máquina de estado con los privilegios mínimos necesarios. Estas IAM funciones generadas automáticamente son válidas para la máquina Región de AWS en la que se crea la máquina de estados.

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Le recomendamos que, al crear políticas de IAM, no incluya caracteres comodín en las políticas. Como práctica recomendada de seguridad, debe reducir el alcance de las políticas en la medida de lo posible. Debe usar políticas dinámicas solo cuando no se conozcan ciertos parámetros de entrada durante el tiempo de ejecución.

Además, los usuarios administradores deben tener cuidado al conceder roles de ejecución a los usuarios que no sean administradores para ejecutar las máquinas de estado. Le recomendamos que incluya políticas de PassRole en los roles de ejecución si va a crear políticas por su cuenta. También

le recomendamos que añada las claves de contexto `aws:SourceARN` y `aws:SourceAccount` en los roles de ejecución.

En este tema

Ejemplos de políticas de IAM para la integración sin servidor de EMR con Step Functions

Ejemplo de política de IAM para CreateApplication

El siguiente es un ejemplo de política de IAM para una máquina de estados con un CreateApplication Estado de la tarea estado.

> **Note**
>
> Debe especificar los CreateServiceLinkedRole permisos en sus políticas de IAM durante la creación de la primera aplicación en su cuenta. A partir de entonces, ya no necesitará añadir este permiso. Para obtener más información CreateServiceLinkedRole, consulte CreateServiceLinkedRolehttps://docs.aws.amazon.com/IAM/latest/APIReference/.

Los recursos estáticos y dinámicos de las siguientes políticas son los mismos.

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"
                }
            }
        }
    ]
}
```

Ejemplo de política de IAM para StartApplication

Recursos estáticos

Los siguientes son ejemplos de políticas de IAM para recursos estáticos cuando se utiliza una máquina de estados con un StartApplication Estado de la tarea estado.

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]]"
             ]
        }
    ]
}
```

Recursos dinámicos

Los siguientes son ejemplos de políticas de IAM para recursos dinámicos cuando se utiliza una máquina de estados con un StartApplication Estado de la tarea estado.

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/*"
             ]
        }
    ]
}
```

Ejemplo de política de IAM para StopApplication

Recursos estáticos

Los siguientes son ejemplos de políticas de IAM para recursos estáticos cuando se utiliza una máquina de estados con un StopApplication Estado de la tarea estado.

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]]"
            ]
        }
    ]
}
```

Recursos dinámicos

Los siguientes son ejemplos de políticas de IAM para recursos dinámicos cuando se utiliza una máquina de estados con un StopApplication Estado de la tarea estado.

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/*"
            ]
        }
    ]
}
```

Ejemplo de política de IAM para DeleteApplication

Recursos estáticos

Los siguientes son ejemplos de políticas de IAM para recursos estáticos cuando se utiliza una máquina de estados con un DeleteApplication Estado de la tarea estado.

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]]"
            ]
        }
    ]
}
```

Recursos dinámicos

Los siguientes son ejemplos de políticas de IAM para recursos dinámicos cuando se utiliza una máquina de estados con un DeleteApplication Estado de la tarea estado.

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/*"
            ]
        }
    ]
}
```

Ejemplo de política de IAM para StartJobRun

Recursos estáticos

Los siguientes son ejemplos de políticas de IAM para recursos estáticos cuando se utiliza una máquina de estados con un StartJobRun Estado de la tarea estado.

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"
                }
            }
        }
    ]
}
```

Recursos dinámicos

Los siguientes son ejemplos de políticas de IAM para recursos dinámicos cuando se utiliza una máquina de estados con un StartJobRun Estado de la tarea estado.

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"
                }
            }
        }
    ]
}
```

Ejemplo de política de IAM para CancelJobRun

Recursos estáticos

Los siguientes son ejemplos de políticas de IAM para recursos estáticos cuando se utiliza una máquina de estados con un CancelJobRun Estado de la tarea estado.

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]]"
            ]
        }
    ]
}
```

Recursos dinámicos

Los siguientes son ejemplos de políticas de IAM para recursos dinámicos cuando se utiliza una máquina de estados con un CancelJobRun [Estado de la tarea](#) estado.

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/*"
            ]
        }
    ]
}
```

Políticas de IAM para Amazon EventBridge

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina estatal. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

`PutEvents`

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:123456789012:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Recursos dinámicos

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

Para obtener más información sobre el uso EventBridge con Step Functions, consulteLlama EventBridge con Step Functions.

Políticas de IAM para AWS Lambda

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener

más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

AWS Step Functions genera una política de IAM basada en la definición de su máquina de estado. En el caso de una máquina de estados con dos estados de AWS Lambda tareas a los que llamar `function1` y`function2`, se debe utilizar una política con `lambda:Invoke` permisos para las dos funciones.

Esto se muestra en el siguiente ejemplo.

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

Políticas de IAM para AWS Elemental MediaConvert

En las siguientes plantillas de ejemplo, se muestra cómo es AWS Step Functions necesario configurar las políticas de IAM en función de los recursos de la definición de su máquina de estados. Puede utilizar la consola de IAM para añadir las políticas de roles que falten. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Debido a que MediaConvert proporciona soporte parcial para el control de acceso a nivel de recursos, debe utilizarla. `"Resource": "*"`

Run a Job (.sync)

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "mediaconvert:CreateJob",
                "mediaconvert:GetJob",
                "mediaconvert:CancelJob"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
               "arn:aws:events:[[region]]:[[accountId]]:rule/
StepFunctionsGetEventsForMediaConvertJobRule"
            ]
        }
    ]
}
```

Request Response

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "mediaconvert:CreateJob"
            ],
            "Resource": "*"
        }
    ]
}
```

Políticas de IAM para AWS Glue

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

AWS Glue no tiene un control basado en los recursos.

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": "*"
        }
    ]
}
```

Políticas de IAM para AWS Glue DataBrew

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

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/*"
            ]
        }
    ]
}
```

Políticas de IAM para Amazon SageMaker

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina estatal. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

> **Note**
>
> Para estos ejemplos, *[[roleArn]]* hace referencia al nombre de recurso de Amazon (ARN) de la función de IAM que se SageMaker utiliza para acceder a los artefactos del modelo y a las imágenes de docker para su implementación en instancias de procesamiento de aprendizaje automático o para trabajos de transformación por lotes. Para obtener más información, consulte Amazon SageMaker Roles.

`CreateTrainingJob`

Recursos estáticos

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"
        }
      }
    }
  ]
}
```

Recursos dinámicos

.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 no creará automáticamente una política para `CreateTransformJob` cuando cree una máquina de estados que se integre con SageMaker. Se debe asociar una política insertada al rol que se ha creado basándose en uno de los siguientes ejemplos de IAM.

Recursos estáticos

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"
        }
      }
    }
  ]
}
```

Recursos dinámicos

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"
        }
      }
    }
```

```
  ]
}
```

Políticas de IAM para Amazon SNS

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
            ]
        }
    ]
}
```

Recursos basados en una ruta o publicación en *`TargetArn`* o *`PhoneNumber`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Políticas de IAM para Amazon SQS

Las siguientes plantillas de ejemplo muestran cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
}
```

Recursos dinámicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

Políticas de IAM para AWS Step Functions

Para una máquina de estado que llame a `StartExecution` para una única ejecución de flujo de trabajo anidado, utilice una política de IAM que limite los permisos a esa máquina de estado.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Para más información, consulte los siguientes temas:

- Uso AWS Step Functions con otros servicios
- Cómo pasar parámetros a una API de servicio
- Gestione AWS Step Functions las ejecuciones como un servicio integrado

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]]"
            ]
        }
```

```
    ]
}
```

Para obtener más información acerca de las ejecuciones de flujos de trabajo anidados, consulte Iniciar ejecuciones de flujo de trabajo desde un estado de tarea.

Políticas de IAM para AWS X-Ray

En las siguientes plantillas de ejemplo, se muestra cómo se AWS Step Functions generan las políticas de IAM en función de los recursos de la definición de su máquina de estados. Para obtener más información, consulte Políticas de IAM para servicios integrados y Patrones de integración de servicios.

Para habilitar el rastreo de X-Ray, necesitará una política de IAM con los permisos adecuados para permitir el rastreo. Si la máquina de estado utiliza otros servicios integrados, es posible que necesite políticas de IAM adicionales. Consulte las políticas de IAM para sus integraciones de servicios específicas.

Al crear una máquina de estado con el rastreo de X-Ray activado, se crea automáticamente una política de IAM.

> **Note**
>
> Si habilita el rastreo de X-Ray para una máquina de estado existente, debe asegurarse de añadir una política con permisos suficientes para habilitar los rastreos de X-Ray.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
            "Resource": [
```

```
                "*"
            ]
        }
    ]
}
```

Para obtener más información sobre el uso de X-Ray con Step Functions, consulte AWS X-Ray y Step Functions.

Actividades o sin tareas

En una máquina de estado donde únicamente haya tareas de `Activity` o no haya ninguna, utilice una política de IAM que deniegue el acceso a todas las acciones y recursos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Para obtener más información acerca del uso de tareas de `Activity`, consulte Actividades.

Políticas de IAM para usar el estado Map Distributed

Al crear flujos de trabajo con la consola de Step Functions, Step Functions puede generar automáticamente políticas de IAM basadas en los recursos de la definición de flujo de trabajo. Estas políticas incluyen los privilegios mínimos necesarios para permitir que el rol de la máquina de estado invoque la acción de la API `StartExecution` para el estado Map Distributed. Estas políticas también incluyen los privilegios mínimos necesarios para que Step Functions pueda acceder a AWS los recursos, como los buckets y objetos de Amazon S3 y las funciones de Lambda. Le recomendamos que incluya únicamente los permisos necesarios en las políticas de IAM. Por ejemplo, si el flujo de trabajo incluye un estado `Map` en modo distribuido, aplique las políticas al bucket y a la carpeta de Amazon S3 específicos que contengan el conjunto de datos.

> **Important**
>
> Si especifica un bucket y un objeto de Amazon S3, o un prefijo, con una ruta de referencia a un par clave-valor existente en la entrada del estado Map Distributed, no olvide actualizar las políticas de IAM para el flujo de trabajo. Limite las políticas hasta los nombres de objeto y bucket a los que se dirige la ruta en tiempo de ejecución.

En este tema:

Ejemplo de política de IAM para ejecutar un estado Map Distributed

Cuando se incluye un estado Map Distributed en los flujos de trabajo, Step Functions necesita los permisos adecuados para permitir que el rol de la máquina de estado invoque la acción de la API `StartExecution` para el estado Map Distributed.

El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios al rol de la máquina de estado para ejecutar el estado Map Distributed.

> **Note**
>
> No olvide reemplazar *`stateMachineName`* por el nombre de la máquina de estado en la que está utilizando el estado Map Distributed. Por ejemplo, `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:*"
    }
  ]
}
```

Ejemplo de política de IAM para redriving de estado Map Distributed

Puede reiniciar las ejecuciones fallidas de flujos de trabajo secundarios en un flujo de trabajo basado en Map Run mediante redriving del flujo de trabajo principal. Un flujo de trabajo principal redriven redrives todos los estados fallidos, incluido el estado Map Distributed. Asegúrese de que la función de ejecución tenga los privilegios mínimos necesarios para poder invocar la acción de la API `RedriveExecution` en el flujo de trabajo principal.

El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios para el rol de la máquina de estado para redriving de un estado Map Distributed.

> **Note**
>
> No olvide reemplazar *stateMachineName* por el nombre de la máquina de estado en la que está utilizando el estado Map Distributed. Por ejemplo, `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:*"
    }
  ]
}
```

Ejemplos de políticas de IAM para leer datos desde conjuntos de datos de Amazon S3

Los siguientes ejemplos de políticas de IAM otorgan los privilegios mínimos necesarios para acceder a sus conjuntos de datos de Amazon S3 mediante las acciones ListObjectsV2 y GetObjectAPI.

Example Política de IAM para el objeto Amazon S3 como conjunto de datos

El siguiente ejemplo muestra una política de IAM que concede los privilegios mínimos necesarios para acceder a los objetos organizados dentro de *processImages* en un bucket de Amazon S3 llamado *myBucket*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example Política de IAM para un archivo CSV como conjunto de datos

En el siguiente ejemplo se muestra una política de IAM que concede los privilegios mínimos necesarios para acceder a un archivo CSV llamado *ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Política de IAM para un inventario de Amazon S3 como conjunto de datos

En el siguiente ejemplo se muestra una política de IAM que concede los privilegios mínimos necesarios para acceder a un informe de inventario de Amazon S3.

```
{
    "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/*"
            ]
        }
    ]
}
```

Ejemplo de política de IAM para escribir datos en un bucket de Amazon S3

El siguiente ejemplo de política de IAM otorga los privilegios mínimos necesarios para escribir los resultados de la ejecución del flujo de trabajo secundario en una carpeta denominada *csvJobs* de un bucket de Amazon S3 mediante la acción PutObject de la API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Permisos de IAM para un bucket de Amazon S3 AWS KMS key cifrado

El estado Map Distributed utiliza cargas multiparte para escribir los resultados de ejecución del flujo de trabajo secundario en un bucket de Amazon S3. Si el bucket se cifra con una clave AWS Key Management Service (AWS KMS), también debe incluir los permisos en la política IAM para realizar las acciones `kms:Decrypt`, `kms:Encrypt` y `kms:GenerateDataKey` sobre la clave Estos permisos son necesarios, ya que Amazon S3 debe descifrar y leer datos de las partes de archivos cifrados antes de finalizar la carga multiparte.

El siguiente ejemplo de política de IAM concede permisos a las acciones `kms:Decrypt`, `kms:Encrypt` y `kms:GenerateDataKey` sobre la clave utilizada para cifrar el bucket de Amazon S3.

```
{
  "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"
```

```
    ]
  }
}
```

Para obtener más información, consulte Cargar un archivo grande en Amazon S3 con cifrado mediante AWS KMS key en el AWS Centro de conocimientos.

Si su usuario o rol de IAM es el Cuenta de AWS mismo que elKMS key, debe tener estos permisos en la política clave. Si el usuario o rol de IAM pertenecen a una cuenta distinta que la KMS key, debe tener los permisos tanto en la política de claves como en el usuario o rol de IAM.

Políticas basadas en etiquetas

Step Functions admite políticas basadas en etiquetas. Por ejemplo, puede restringir el acceso a todos los recursos de Step Functions que incluyan una etiqueta con la clave `environment` y el valor `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"}
            }
        }
    ]
}
```

Esta política denegará (`Deny`) la posibilidad de eliminar máquinas de estado o actividades, detener ejecuciones y añadir o eliminar nuevas etiquetas para todos los recursos que se han etiquetado como `environment/production`.

En el caso de la autorización basada en etiquetas, los recursos de ejecución de máquinas de estado, como se muestra en el siguiente ejemplo, heredan las etiquetas asociadas a una máquina de estado.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Cuando llamas a DescribeExecutionotras API en las que especificas el ARN del recurso de ejecución, Step Functions utiliza las etiquetas asociadas a la máquina de estados para aceptar o denegar la solicitud mientras realiza la autorización basada en etiquetas. Esto ayuda a permitir o denegar el acceso a las ejecuciones de máquina de estado en el nivel de las máquinas de estado.

Para obtener más información sobre el etiquetado, consulte lo siguiente:

- Etiquetado en Step Functions
- Control del acceso mediante etiquetas de IAM

Solución de problemas de AWS Step Functions identidad y acceso

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con Step Functions e IAM.

Temas

- No tengo autorización para realizar una acción en Step Functions
- No estoy autorizado a realizar tareas como: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Step Functions

No tengo autorización para realizar una acción en Step Functions

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `states:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

En este caso, la política de Mateo se debe actualizar para permitirle acceder al recurso *my-example-widget* mediante la acción `states:GetWidget`.

Si necesita ayuda, póngase en contacto con su AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, se deben actualizar las políticas a fin de permitirle pasar un rol a Step Functions.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Step Functions. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Step Functions

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para obtener información acerca de si Step Functions admite estas características, consulte ¿Cómo AWS Step Functions funciona con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer la diferencia entre usar roles y políticas basadas en recursos para el acceso entre cuentas, consulte el tema Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registro y supervisión

Para obtener información sobre el inicio de sesión y la supervisión, consulte la sección AWS Step Functions. Registro y monitorización

Validación de conformidad para AWS Step Functions

Los auditores externos evalúan la seguridad y el cumplimiento AWS Step Functions como parte de varios programas de AWS cumplimiento. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros.

Para obtener una lista de AWS los servicios incluidos en el ámbito de los programas de cumplimiento específicos, consulte los AWS servicios incluidos en el ámbito de aplicación por programa de cumplimiento y . Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad en el ámbito de la conformidad al usar Step Functions viene determinada por la confidencialidad de los datos, los objetivos de conformidad de su empresa y las leyes y regulaciones aplicables. AWS proporciona los siguientes recursos para ayudarlo con los requisitos de conformidad:

- Guías de inicio rápido sobre : estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en la seguridad y el cumplimiento. AWS
- Diseño de arquitectura para la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: este documento técnico describe cómo las empresas pueden utilizar AWS para crear aplicaciones compatibles con la HIPAA.
- AWS Recursos de cumplimiento Recursos de : esta colección de libros de trabajo y guías puede aplicarse a su sector y ubicación.
- Evaluación de los recursos con las reglas de la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Este AWS servicio proporciona una visión integral del estado de su seguridad AWS que le ayuda a comprobar su conformidad con los estándares y las mejores prácticas del sector de la seguridad.

Resiliencia en AWS Step Functions

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS

Además de la infraestructura AWS global, Step Functions ofrece varias funciones que ayudan a respaldar sus necesidades de respaldo y resiliencia de datos.

Seguridad de la infraestructura en AWS Step Functions

Como servicio gestionado, AWS Step Functions está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las

mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a Step Functions través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad de seguridad de IAM principal. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Puedes llamar a las operaciones de la AWS API desde cualquier ubicación de la red, pero Step Functions no admite políticas de acceso basadas en los recursos, que pueden incluir restricciones basadas en la dirección IP de origen. Además, puede utilizar las políticas de Step Functions para controlar el acceso desde puntos de enlace de Amazon Virtual Private Cloud (Amazon VPC) o desde VPC específicas. En efecto, esto aísla el acceso a la red a un Step Functions recurso determinado únicamente de la VPC específica de la red. AWS

Análisis de configuración y vulnerabilidad en AWS Step Functions

La configuración y los controles de TI son una responsabilidad compartida entre usted AWS y usted, nuestro cliente. Para obtener más información, consulte el modelo de responsabilidad AWS compartida.

Migración de cargas de trabajo de Step Functions AWS Data Pipeline a Step Functions

AWS lanzó el AWS Data Pipeline servicio en 2012. En ese momento, los clientes querían un servicio que les permitiera usar una serie de opciones de computación para mover datos entre diferentes orígenes de datos. A medida que las necesidades de transferencia de datos cambiaban con el tiempo, también lo han hecho las soluciones para esas necesidades. Ahora tiene la opción de elegir la solución que mejor se adapte a sus requisitos empresariales. Por ejemplo, puede hacer lo siguiente:

- Usar Step Functions para orquestar flujos de trabajo entre varios Servicios de AWS.
- Usar Amazon Managed Workflows para Apache Airflow (Amazon MWAA) para gestionar la orquestación de flujos de trabajo de Apache Airflow.
- Se utiliza AWS Glue para ejecutar y organizar aplicaciones de Apache Spark.

Puede migrar los casos de uso típicos AWS Data Pipeline a Step Functions o Amazon MWAA. AWS Glue La opción que elija depende de su carga de trabajo actual en AWS Data Pipeline. En este tema se explica cómo migrar de Step Functions AWS Data Pipeline a Step Functions.

Temas

Migración de cargas de trabajo de AWS Data Pipeline

Step Functions es un servicio de orquestación sin servidor donde se pueden crear flujos de trabajo para aplicaciones esenciales desde el punto de vista empresarial. Con Workflow Studio de Step Functions, puede crear flujos de trabajo e integrarlos con más de 11 000 acciones de API de más de 250 Servicios de AWS. Esto incluye Servicios de AWS Amazon EMR y Amazon DynamoDB. AWS Lambda También puede usar Step Functions para orquestar las canalizaciones de procesamiento de datos, gestionar los errores y trabajar con los límites de regulación en Servicios de AWS subyacentes. Puede crear flujos de trabajo que procesen y publiquen modelos de machine learning,

orquestar microservicios y gestionar flujos de trabajo de extracción, transformación y carga (ETL) con AWS Glue. También puede crear flujos de trabajo automatizados y de larga duración para aplicaciones que requieren la interacción humana.

Step Functions es un servicio totalmente gestionado por AWS. Esto significa que AWS gestiona tareas por usted, como el mantenimiento de la infraestructura, la aplicación de parches a los trabajadores y la gestión de las actualizaciones de las versiones del sistema operativo.

Cuando su caso de uso cumpla las siguientes condiciones, le recomendamos que migre de Step Functions AWS Data Pipeline a:

- Es preferible un servicio de orquestación de flujos de trabajo sin servidor y de alta disponibilidad.
- Es necesaria una solución que cobre en la granularidad de la ejecución de una sola tarea.
- Sus cargas de trabajo implican la organización de tareas para muchas otras Servicios de AWS, como Amazon EMR, Lambda AWS Glue o DynamoDB.
- Necesita una solución de bajo código con un diseñador visual para la creación de flujos de trabajo. drag-and-drop Esta solución no debería requerir el aprendizaje de conceptos de programación complejos y desconocidos.
- Necesita un servicio que se integre con más de 250 Servicios de AWS que abarquen más de 11 000 acciones de API. Este servicio también debe integrarse con servicios y actividades personalizados externos a. AWS

Asignación de conceptos entre Step Functions y AWS Data Pipeline

AWS Data Pipeline y Step Functions comparten algunos conceptos comunes. Por ejemplo, para definir tus flujos de trabajo, utilizas el formato JSON tanto en Step Functions como AWS Data Pipeline en Step Functions. En Step Functions, se utiliza Lenguaje de estados de Amazon, que es un lenguaje estructurado basado en JSON. Use Amazon States Language (ASL) para definir sus flujos de trabajo y cambiar entre las representaciones textuales y visuales del flujo de trabajo. Este formato basado en JSON ayuda a simplificar el almacenamiento de los flujos de trabajo en una herramienta de control de origen. También contribuye a gestionar varias versiones de los flujos de trabajo, controlar su acceso o automatizar su orquestación con métodos de CI/CD.

En la siguiente tabla se describe la asignación entre los principales conceptos que se utilizan en ambos servicios. La columna de conceptos de Data Pipeline de la izquierda muestra los conceptos

de Step Functions AWS Data Pipeline, mientras que la columna de conceptos de Step Functions, de la derecha, muestra los conceptos equivalentes de Step Functions.

Conceptos de canalización de datos	Conceptos de Step Functions
Canalizaciones	Flujos de trabajo
Definición de la canalización	Lenguaje de estados de Amazon (ASL)
Actividades	Estados y Estado de la tarea
Instancias	Ejecuciones
Attempts	Captadores y reintentadores
Calendario de canalización	• Ejecuciones con Amazon EventBridge Scheduler • Eventos activados a través EventBridge de Pipes
Expresiones y funciones de canalizaciones	• Funciones intrínsecas • Funciones de Lambda mediante la integración de servicios

Proyectos de muestra de Step Functions

Para obtener una introducción a Step Functions, consulte el siguiente vídeo:

Primeros pasos con la AWS Step Functions orquestación de servicios

En la siguiente lista se describen algunos proyectos de muestra que implementan los casos de uso de AWS Data Pipeline más comunes con Step Functions. Puedes usar estos proyectos de ejemplo como referencia para migrar de Step Functions AWS Data Pipeline a Step Functions. También puede utilizarlos como referencia para crear sus propios flujos de trabajo e integrarlos con los Servicios de AWS compatibles según su caso de uso.

- Administrar un trabajo de Amazon EMR
- Ejecute un trabajo de procesamiento de datos en Amazon EMR sin servidor

- Ejecución de trabajos de Hive/Pig/Hadoop
- Consulte conjuntos de datos de gran tamaño (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)
- Ejecutar flujos de trabajo de ETL/ELT con Amazon Redshift
- Organizar rastreadores AWS Glue
- Ejecutar un script de intérprete de comandos con Step Functions

Para obtener más información acerca de Step Functions, consulte los temas y recursos siguientes:

- Tutoriales de Step Functions
- Proyectos de muestra para Step Functions
- El taller de AWS Step Functions

Comparación de precios

AWS Data Pipeline su precio se basa en el número de canalizaciones y su nivel de uso. Las actividades que se ejecutan más de una vez al día (frecuencia alta) tienen un precio de 1 USD al mes por actividad. Las actividades que se ejecutan una vez al día o menos (frecuencia baja) tienen un precio de 0,60 USD al mes por actividad. Las canalizaciones inactivas tienen un precio de 1 USD por canalización. Para obtener más información acerca de los precios, consulte la página Precios de AWS Data Pipeline.

Step Functions tiene dos tipos de flujos de trabajo: estándar y rápido. Cada tipo de flujo de trabajo tiene un modelo de precios diferente. Esta comparación se basa en el flujo de trabajo estándar, ya que es el que mejor se adapta a los casos de uso comunes de AWS Data Pipeline. Los flujos de trabajo estándar tienen un precio de 0,025 USD por cada 1000 transiciones de estado. Las máquinas de estado inactivas no tienen costo; solo se paga por lo que se usa. Para obtener más información acerca de los precios, consulte la página Precios de AWS Step Functions.

Solución de problemas

Si surgen problemas a la hora de trabajar con Step Functions, utilice los siguientes recursos.

Temas

Solución de problemas generales

No puedo crear una máquina de estado.

Es posible que el rol de IAM asociado a la máquina de estado no tenga permisos suficientes. Compruebe los permisos del rol de IAM, incluidos los de las tareas de integración de servicios de AWS y el registro de CloudWatch y X-Ray. Se requieren permisos adicionales para los estados de las tareas de estado de `.sync`.

No puedo utilizar una JsonPath para hacer referencia a la salida de la tarea anterior.

Para una JsonPath, la clave JSON debe terminar en `.$`. Esto significa que una JsonPath solo se puede utilizar en un par de clave-valor. Si desea utilizar una JsonPath en otros lugares, como una matriz, puede utilizar funciones intrínsecas. Por ejemplo, puede utilizar algo similar a lo siguiente:

Salida de tarea A:

```
{
    "sample": "test"
}
```

Tarea B:

```
{
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> Utilice el simulador de flujo de datos de la consola de Step Functions para probar la sintaxis de las rutas JSON, comprender mejor cómo se manipulan los datos dentro de un estado y ver cómo se transfieren los datos entre los estados.

Se produjo un retraso en las transiciones de estado.

Para los flujos de trabajo estándar hay un límite en el número de transiciones de estado. Cuando supera el límite de transiciones de estado, Step Functions retrasa las transiciones de estado hasta que se llene el bucket de la cuota. La limitación de las transiciones de estado se puede supervisar revisando la métrica de `ExecutionThrottled` en la sección Métricas de ejecución de la página Métricas de CloudWatch.

Cuando inicio nuevas ejecuciones de flujos de trabajo estándar se produce un error de **`ExecutionLimitExceeded`**.

Step Functions tiene un límite de 1 000 000 de ejecuciones abiertas para cada Cuenta de AWS en cada Región de AWS. Si supera este límite, Step Functions lanza un error de `ExecutionLimitExceeded`. Este límite no se aplica a los flujos de trabajo rápidos. Puede utilizar las siguientes Matemáticas de CloudWatch Metrics en la Guía del usuario de Amazon CloudWatch para aproximar el número de ejecuciones abiertas: `ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`.

Un error en una rama en estado paralelo provoca un error en toda la ejecución.

Este es el comportamiento esperado. Para evitar errores al utilizar un estado paralelo, configure Step Functions para detectar los errores lanzados desde cada rama.

Solución de problemas de integración de servicios

Mi trabajo está finalizado en el servicio descendente, pero en Step Functions el estado de la tarea sigue siendo "En curso" o su finalización se retrasa.

Para los patrones de integración de servicios de `.sync`, Step Functions utiliza reglas de EventBridge, API descendentes o una combinación de ambas para detectar el estado de las tareas descendentes. Para algunos servicios, Step Functions no crea reglas de EventBridge para su supervisión. Por ejemplo, para la integración de servicios de AWS Glue, en lugar de utilizar las reglas de EventBridge, Step Functions realiza una llamada `glue:GetJobRun`. Dada la frecuencia de las llamadas a la API, hay una diferencia entre la finalización de tareas descendentes y el tiempo de finalización de tareas de Step Functions. Step Functions requiere permisos de IAM para gestionar las reglas de EventBridge y realizar llamadas al servicio descendente. Para obtener más información sobre cómo la falta de permisos en su rol de ejecución puede afectar a la finalización de las tareas, consulte Permisos adicionales para las tareas que utilizan el patrón de ejecución de un trabajo.

Quiero devolver una salida de JSON desde una ejecución de máquina de estado anidada.

Hay dos integraciones de servicios síncronos de Step Functions para Step Functions: `startExecution.sync` y `startExecution.sync:2`. Ambos esperan a que se complete la máquina de estado anidada, pero devuelven diferentes formatos `Output`. Puede utilizar `startExecution.sync:2` para devolver una salida de JSON en `Output`.

No puedo invocar una función de Lambda desde otra cuenta.

Acceder a la función de Lambda con compatibilidad entre cuentas

Si el acceso entre cuentas a los recursos de AWS está disponible en su región, utilice el siguiente método para invocar una función de Lambda desde otra cuenta.

Para invocar un recurso entre cuentas en los flujos de trabajo, haga lo siguiente:

1. Cree un rol de IAM en la cuenta de destino que contiene el recurso. Esta función otorga a la cuenta de origen, que contiene la máquina de estado, permisos para acceder a los recursos de la cuenta de destino.

2. En la definición del estado `Task`, especifique el rol de IAM de destino que asumirá la máquina de estado antes de invocar el recurso entre cuentas.
3. Modifique la política de confianza del rol de IAM de destino para permitir que la cuenta de origen asuma esta función temporalmente. La política de confianza debe incluir el nombre de recurso de Amazon (ARN) de la máquina de estado definida en la cuenta de origen. Además, defina los permisos adecuados en el rol de IAM de destino para llamar al recurso de AWS.
4. Actualice el rol de ejecución de la cuenta de origen para incluir el permiso necesario para asumir el rol de IAM de destino.

Para ver un ejemplo, consulte Tutorial: Acceso a recursos multicuenta AWS.

> **Note**
>
> Puede configurar la máquina de estado para que asuma un rol de IAM para acceder a los recursos desde varias Cuentas de AWS. No obstante, una máquina de estado solo puede asumir un rol de IAM en un momento dado.

Para ver un ejemplo de una definición de estado de `Task` que especifica un recurso multicuenta, consulte Ejemplos del campo Credentials de estado de tarea.

Acceder a la función de Lambda sin compatibilidad entre cuentas

Si el acceso entre cuentas a los recursos de AWS no está disponible en su región, utilice el siguiente método para invocar una función de Lambda desde otra cuenta.

En el campo `Resource` del estado de `Task`, utilice `arn:aws:states:::lambda:invoke` y pase el `FunctionArn` en parámetros. El rol de IAM asociado a la máquina de estado debe tener los permisos adecuados para invocar funciones de Lambda entre cuentas: `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
      }
   }
}
```

No puedo ver los tokens de tarea transferidos desde los estados **`.waitForTaskToken`**.

En el campo `Parameters` del estado de `Task`, debes pasar un token de tarea. Por ejemplo, puede utilizar algo similar al código siguiente.

```
{
   "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**
>
> Puede intentar utilizar `.waitForTaskToken` con cualquier acción de la API. Sin embargo, algunas API no tienen ningún parámetro adecuado.

Solución de problemas de actividades

La ejecución de mi máquina de estado está bloqueada en un estado de actividad.

El estado de una tarea de actividad no se inicia hasta que se sondea un token de tarea mediante la acción de la API GetActivityTask. Se recomienda añadir un tiempo de espera de nivel de tarea para evitar que la ejecución se bloquee. Para obtener más información, consulte Utilizar tiempos de espera para evitar las ejecuciones bloqueadas.

Si su máquina de estado se bloquea en el evento ActivityScheduled, indica que la flota de procesos de trabajo de actividad tiene problemas o está infradimensionada. Debe supervisar la métrica de `ActivityScheduleTime` CloudWatch y configurar una alarma cuando aumente ese tiempo. Sin embargo, para agotar el tiempo de espera de ejecuciones de máquina de estado bloqueadas en las que el estado `Activity` no pase al estado `ActivityStarted`, defina un tiempo de espera de nivel de máquina de estado. Para ello, especifique un campo `TimeoutSeconds` al principio de la definición de máquina de estado, fuera del campo `States`.

Mi proceso de trabajo de actividad agota el tiempo de espera mientras espera un token de tarea.

Los empleados utilizan la acción de la API GetActivityTask para recuperar una tarea con el ARN de actividad especificado cuya ejecución ha sido programada por una máquina de estado en ejecución. `GetActivityTask` inicia un sondeo largo para que el servicio mantenga abierta la conexión HTTP y responda en cuanto haya una tarea disponible. El tiempo máximo que el servicio retiene la solicitud antes de responder es de 60 segundos. Si no hay ninguna tarea disponible en 60 segundos, el sondeo devuelve un `taskToken` con una cadena nula. Para evitar este tiempo de espera, configure un socket del lado del cliente con un tiempo de espera de al menos 65 segundos en el SDK de AWS o en el cliente que utilice para realizar la llamada a la API.

Solución de problemas de flujos de trabajo rápidos

Se agota el tiempo de espera de mi aplicación antes de recibir una respuesta de una llamada a la API **`StartSyncExecution`**.

Configure un tiempo de espera del socket del lado del cliente en el SDK de AWS o en el cliente que utilice para realizar la llamada a la API. Para recibir una respuesta, el tiempo de espera debe tener un valor superior a la duración de las ejecuciones de flujos de trabajo rápidos.

No puedo ver el historial de ejecuciones para solucionar los errores de flujos de trabajo rápidos.

Los flujos de trabajo rápidos no registran el historial de ejecuciones en AWS Step Functions. En su lugar, debe activar el registro de CloudWatch. Una vez activado el registro, puede utilizar las consultas de CloudWatch Logs Insights para revisar las ejecuciones de Express Workflow. También puede ver el historial de ejecuciones de flujos de trabajo rápidos en la consola de Step Functions si selecciona el botón Habilitar de la pestaña Ejecuciones. Para obtener más información, consulte Visualización y depuración de ejecuciones en la consola de Step Functions.

Para mostrar las ejecuciones en función de su duración:

```
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
```

Para mostrar ejecuciones con error y canceladas:

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

Información relacionada

En la tabla siguiente se enumeran todos los recursos relacionados que podrían resultarle útiles cuando trabaje con este servicio.

Recurso	Descripción
Referencia de la API de AWS Step Functions	Descripciones de las acciones, parámetros y tipos de datos de la API, así como una lista de errores que el servicio devuelve.
Referencia de línea de comandos de AWS Step Functions	Descripciones de los comandos de la AWS CLI que puede utilizar para trabajar con AWS Step Functions.
Información del producto de Step Functions	Página web principal con información acerca de Step Functions.
Foros de debate	Un foro de la comunidad para desarrolladores donde se tratan aspectos técnicos relacionados con Step Functions y otros servicios de AWS.
Información de AWS Support	Página web principal para obtener información acerca de AWS Support, un canal de soporte individualizado y de respuesta rápida que le ayudará a crear y ejecutar aplicaciones en servicios de infraestructura de AWS.

Lanzamientos recientes de características

En la siguiente tabla se muestran las regiones en las que están disponibles las nuevas características de Step Functions.

Fecha de lanzamiento	Nombre de la característica	Regiones disponibles
26 de noviembre de 2023	Invocar puntos de conexión HTTPS públicos y probar estados individuales	• EE. UU. Este (Norte de Virginia) us-east-1 • EE. UU. Oeste (Oregón) us-west-2 • EE. UU. Este (Ohio) us-east-2 • UE (Irlanda) eu-west-1 • UE (Fráncfort) eu-central-1 • UE (Estocolmo) eu-north-1 • Asia-Pacífico (Sídney) ap-southeast-2 • Asia-Pacífico (Tokio) ap-northeast-1 • Asia-Pacífico (Singapur) ap-southeast-1
15 de noviembre de 2023	Redrive ejecuciones	Para obtener una lista completa de las Regiones de AWS en las que está disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.
12 de octubre de 2023	Integración optimizada para Amazon EMR Serverless	Para obtener una lista completa de las Regiones de AWS en las que está

Fecha de lanzamiento	Nombre de la característica	Regiones disponibles
		disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.
7 de septiembre de 2023	Control de errores mejorado	Para obtener una lista completa de las Regiones de AWS en las que está disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.
31 de agosto de 2023	Mejoras en Workflow Studio para una experiencia de creación optimizada	Para obtener una lista completa de las Regiones de AWS en las que está disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.
22 de junio de 2023	Versiones y alias	Para obtener una lista completa de las Regiones de AWS en las que está disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.

Fecha de lanzamiento	Nombre de la característica	Regiones disponibles
16 de junio de 2023	Nuevas integraciones del SDK de AWS	Para obtener una lista completa de las Regiones de AWS en las que está disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.
1 de diciembre de 2022	Orqueste flujos de trabajo paralelos a gran escala para el procesamiento de datos con el estado Distributed Map	Para obtener una lista completa de las Regiones de AWS en las que está disponible esta característica, consulte las opciones en la lista desplegable Región de la página titulada Servicios de AWS por región.

Historial de documentos

En esta sección se indican los cambios principales en la Guía par desarrolladores de AWS Step Functions .

Cambio	Descripción	Fecha de modificación
Actualizaciones	AWS actualizaciones de políticas gestionadas: nuevo permiso: `states:ValidateStateMachineDefinition` Se agregó información sobre el nuevo permiso para comprobar la sintaxis de una máquina de estados que usted proporcione. Para obtener más información, consulte AWS políticas gestionadas para AWS Step Functions.	29 de abril de 2024
Nueva característica	Step Functions añade una integración optimizada para AWS Elemental MediaConvert AWS Elemental MediaConvert proporciona una transcodificación de archivos de audio y vídeo apta para la radiodifusión, que los clientes pueden automatizar con código para adaptarla a sus flujos de trabajo multimedia. Gracias a la integración optimizada de AWS Step Functions In MediaConvert, ahora es posible orquestar con la herramienta visual de bajo código Workflow Studio. Para obtener más información, consulte la documentación de Manage AWS Elemental MediaConvert with Step Functions.	12 de abril de 2024
Actualizaciones	AWS actualizaciones de políticas gestionadas: actualización de una política existente: `AWSStepFunctionsReadOnlyAccess` Se agregó información sobre los nuevos permisos de solo lectura para etiquetas, mapas distribuidos y versiones	2 de abril de 2024

Cambio	Descripción	Fecha de modificación
	y alias. Para obtener más información, consulte AWS políticas gestionadas para AWS Step Functions.	
Actualizaciones	Step Functions añade compatibilidad con las métricas de Open Workflow Con las métricas de flujos de trabajo abiertos, ahora tiene visibilidad a nivel de cuenta del número de flujos de trabajo estándar en curso, así como del límite de flujos de trabajo abiertos. Puede gestionar las cargas de trabajo en todos los flujos de trabajo, independientemente de cómo se hayan iniciado, para garantizar que las operaciones del flujo de trabajo sean fluidas. Puede configurar CloudWatch alarmas para supervisar sus flujos de trabajo y recibir alertas de forma proactiva a medida que se acerca a sus límites. Una vez que recibas una alerta, puedes gestionar tus flujos de trabajo de forma eficaz tomando medidas como detener flujos de trabajo específicos o solicitar un aumento del límite. Las métricas de flujo de trabajo abiertas están disponibles CloudWatch para su uso en los flujos de trabajo estándar sin necesidad de configuración adicional. Para obtener más información, consulte Métricas de ejecución.	29 de febrero de 2024
Actualizaciones	Adiciones y actualizaciones de la integración de servicios . Para ver la lista de integraciones de AWS SDK nuevas y actualizadas, consulteRegistro de cambios para las integraciones AWS de SDK compatibles. Para ver la lista completa de servicios, consulteIntegraciones de servicios AWS de SDK compatibles.	18 de enero de 2024

Cambio	Descripción	Fecha de modificación
Nueva característica	Utilice Workflow Studio en Application Composer para crear flujos de trabajo sin servidor mediante plantillas de AWS CloudFormation. Para obtener más información, consulte Uso de Workflow Studio en Application Composer.	27 de noviembre de 2023
Nueva característica	Step Functions permite ahora invocar directamente puntos de conexión HTTPS públicos y probar estados individua les mediante una nueva API de estado de prueba. Para obtener más información, consulte: • Llamada a API de terceros • Uso TestState de la API para probar un estado	26 de noviembre de 2023
Nueva característica	Step Functions ahora se integra con Amazon Bedrock Para obtener más información, consulte los temas siguientes: • Llamada a Amazon Bedrock con Step Functions • Permisos de IAM para Amazon Bedrock • Encadenamiento de mensajes de IA con Amazon Bedrock • Uso AWS Step Functions con otros servicios	26 de noviembre de 2023
Nueva característica	Step Functions ahora permite redrive ejecuciones de flujos de trabajo de tipo estándar desde su punto de error. Para obtener más información, consulte Redriving de ejecucion es y Redriving de Map Runs.	15 de noviembre de 2023
Actualización exclusiva de la documentación	Se ha publicado un nuevo tema que explica cómo ejecutar máquinas de estado según una programación utilizand o Amazon EventBridge Scheduler. Para obtener más información, consulte Uso de Programador de Amazon EventBridge con AWS Step Functions.	16 de octubre de 2023

Cambio	Descripción	Fecha de modificación
Nueva característica	Step Functions ahora se integra con Amazon EMR Serverless Para obtener más información, consulte los temas siguientes: • Llamada a Amazon EMR Serverless con Step Functions • Ejecutar un trabajo de EMR Serverless • Integraciones optimizadas para Step Functions • Uso AWS Step Functions con otros servicios	12 de octubre de 2023
Actualización exclusiva de la documentación	Se ha añadido información sobre cómo ejecutar máquinas de estado según una programación utilizando Amazon EventBridge Scheduler. Para obtener más información, consulte Uso de un programador de EventBridge.	5 de octubre de 2023
Actualización	Se ha reorganizado y actualizado los temas de estado Distributed Map para mayor claridad y brevedad y para establecer una hoja de ruta clara para los nuevos usuarios. Para obtener más información, consulte Uso del estado Map en modo distribuido para orquestar cargas de trabajo paralelas a gran escala.	6 de octubre de 2023
Correcciones	Se han corregido ejemplos de código en un tutorial para usar AWS CDK v2. Para obtener más información, consulte Creación de una máquina de estado Lambda para Step Functions mediante AWS CDK.	19 de septiembre de 2023
Actualización	Se ha añadido información sobre las funciones mejoradas de control de errores que Step Functions ha introducido para identificar los errores con claridad e implementar los reintentos con más control. Para obtener más información, consulte Fail y Reintento después de un error.	7 de septiembre de 2023

Cambio	Descripción	Fecha de modificación
Actualización	Step Functions ha añadido mejoras a Workflow Studio para simplificar la experiencia de creación de flujos de trabajo. Para obtener más información, consulte AWS Step Functions Estudio de flujo de trabajo.	31 de agosto de 2023
Actualización exclusiva de la documentación	Se ha añadido información sobre el doble del número de métricas real registrado para la métrica `ExecutionsStarted`. Para obtener más información, consulte Métricas que informan de un recuento.	25 de julio de 2023
Actualización exclusiva de la documentación	Step Functions ha añadido dos nuevos proyectos de muestra que ilustran los siguientes casos de uso comunes del estado Distributed Map: • Procesamiento de un archivo CSV • Procesamiento de datos en un bucket de Amazon S3	17 de julio de 2023
Actualización exclusiva de la documentación	Se ha publicado un nuevo tema sobre la implementación de máquinas de estados con Terraform. Para obtener más información, consulte Implementar máquinas de estado con Terraform.	5 de julio de 2023
Actualización de la documentación únicamente	Se han actualizado los siguientes procedimientos para que coincidan con los cambios en la EventBridge interfaz de Amazon. • Enrutar un evento de Step Functions a EventBridge • Iniciar ejecución de una máquina de estado en respuesta a eventos de Amazon S3	26 de junio de 2023

Cambio	Descripción	Fecha de modificación
Nueva característica	Step Functions ahora ofrece la posibilidad de crear varias versiones y alias de máquinas de estado para mejorar la resiliencia a la vez que se implementan flujos de trabajo sin servidor. Para obtener más información, consulte Gestión de implementaciones continuas con versiones y alias.	22 de junio de 2023
Actualización exclusiva de la documentación	Se ha mejorado la descripción de los campos `TimeoutSeconds` y `HeartbeatSeconds` para describir en qué se diferencian unos de otros. Para obtener más información, consulte Campos de estado de tarea.	22 de junio de 2023
Actualización exclusiva de la documentación	Se ha publicado una nueva sección que describe cómo aplanar una matriz de matrices devueltas normalmente como resultado para los estados Parallel y Map. Para obtener más información, consulte Aplanamiento de una matriz de matrices.	20 de junio de 2023
Actualización	Step Functions ha ampliado el soporte para las integraciones AWS del SDK al añadir siete Servicios de AWS y 468 nuevas acciones de API. Para obtener más información, consulte Integraciones de servicios AWS de SDK compatibles y Registro de cambios para las integraciones AWS de SDK compatibles.	16 de junio de 2023
Actualización de la documentación únicamente	Se ha publicado un nuevo tema Regiones de AWS en el que se enumeran las funciones de Step Functions lanzadas recientemente. Para obtener más información, consulte Lanzamientos recientes de características.	16 de junio de 2023

Cambio	Descripción	Fecha de modificación
Actualización de la documentación únicamente	Step Functions ahora incluye una sección sobre AWS User Notifications, y Servicio de AWS que actúa como una ubicación central para tus AWS notificaciones en el AWS Management Console. Para obtener más información, consulte Uso de AWS User Notifications con Step Functions.	4 de mayo de 2023
Actualización exclusiva de la documentación	Se ha añadido una nueva sección en la que se explican los permisos necesarios para escribir los resultados de la ejecución del flujo de trabajo secundario en un bucket de Amazon S3 cifrado con una clave AWS Key Management Service (AWS KMS). Para obtener más información, consulte Permisos de IAM para un bucket de Amazon S3 AWS KMS key cifrado.	29 de abril de 2023
Actualización de la documentación únicamente	Se ha añadido un tema nuevo que explica la característica Simulador de flujo de datos. Para obtener más información, consulte Simulador de flujo de datos.	14 de abril de 2023
Actualización de cuotas	Se ha añadido información sobre la cuota predeterminada de 1000 para las Map Runs abiertas en cada cuenta. Para obtener más información, consulte Cuotas relacionadas con las cuentas.	5 de abril de 2023
Actualización de la documentación únicamente	Se agregó un tema que describe cuándo migrar las AWS Data Pipeline cargas de trabajo a Step Functions. En este tema también se proporciona una lista de ejemplos que explican cómo realizar la migración. Para obtener más información, consulte Migración de cargas de trabajo de Step Functions AWS Data Pipeline a Step Functions.	30 de marzo de 2023

Cambio	Descripción	Fecha de modificación
Actualización exclusiva de la documentación	Se ha añadido una nota sobre la falta de disponibilidad del rastreo de X-Ray para el estado Distributed Map. Para obtener más información, consulte AWS X-Ray y Step Functions.	21 de marzo de 2023
Actualización exclusiva de la documentación	Se ha añadido información acerca de cómo Step Functions gestiona la autorización basada en etiquetas. Para obtener más información, consulte Etiquetado en Step Functions y Políticas basadas en etiquetas.	15 de marzo de 2023
Actualización exclusiva de la documentación	Se agregó información sobre cómo Step Functions analiza los archivos CSV utilizados como entrada en el estado Distributed Map. Para obtener más información, consulte Archivo CSV en un bucket de Amazon S3.	14 de marzo de 2023
Actualización exclusiva de la documentación	Se ha añadido información sobre cómo Step Functions gestiona las invocaciones entre cuentas para el patrón Ejecutar un trabajo (.sync). Para obtener más información, consulte Ejecutar un trabajo (.sync).	1 de marzo de 2023
Actualización exclusiva de la documentación	Reduzca el periodo de retención del historial de las ejecuciones de sus flujos de trabajo completados de 90 a 30 días. Para obtener más información acerca de cómo ajustar el periodo de retención, consulte Garantías de ejecución y Cuotas relacionadas con ejecuciones de máquinas de estado.	21 de febrero de 2023
Actualización	Step Functions ha ampliado el soporte para las integraciones de AWS SDK al añadir 35 AWS servicios y 1100 nuevas acciones de API. Para obtener más información, consulte Integraciones de servicios AWS de SDK compatibles y Registro de cambios para las integraciones AWS de SDK compatibles.	17 de febrero de 2023

Cambio	Descripción	Fecha de modificación
Actualización exclusiva de la documentación	Publicación de una serie de tutoriales de introducción que le guían por el proceso de crear un flujo de trabajo para una solicitud de tarjetas de crédito con Step Functions. Para obtener más información, consulte Empezar con AWS Step Functions.	30 de diciembre de 2022
Nueva característica	Step Functions añade compatibilidad para orquestar flujos de trabajo paralelos a gran escala para el procesamiento de datos mediante un nuevo modo distribuido para el estado `Map`. Para obtener más información, consulte Uso del estado Map en modo distribuido para orquestar cargas de trabajo paralelas a gran escala.	1 de diciembre de 2022
Nueva característica	Step Functions ahora admite el acceso a AWS los recursos multicuenta configurados en otras cuentas. Para obtener más información, consulte • Acceder a los recursos de otras partes Cuentas de AWS de sus flujos de trabajo • Tutorial: Acceso a recursos multicuenta AWS • Task state	18 de noviembre de 2022
Actualización	Step Functions ahora ofrece una nueva experiencia de consola para visualizar y depurar las ejecuciones de flujos de trabajo rápidos. Para obtener más información, consulte: • Ejecuciones de flujos de trabajo estándar y rápidos en la consola • Visualización y depuración de ejecuciones en la consola de Step Functions	18 de octubre de 2022

Cambio	Descripción	Fecha de modificación
Actualización	Se ha añadido compatibilidad para especificar opcionalmente el parámetro `ExecutionRoleArn` mientras se utilizan las API de `addStep` y `addStep.sync` para la integración optimizada de servicios de Amazon EMR. Para obtener más información, consulte Llamar a Amazon EMR con Step Functions.	20 de septiembre de 2022
Actualización exclusiva de la documentación	Se ha añadido un tema nuevo que ofrece recomendaciones sobre cómo optimizar los costos al crear flujos de trabajo sin servidor mediante Step Functions. Para obtener más información, consulte Optimización de costes mediante flujos de trabajo rápidos.	15 de septiembre de 2022

Cambio	Descripción	Fecha de modificación
Actualización	Step Functions añade compatibilidad con 14 nuevas funciones intrínsecas para realizar tareas de procesamiento de datos, como la manipulación de matrices, la codificación y decodificación de datos, los cálculos de hash, la manipulación de datos JSON, las operaciones con funciones matemáticas y la generación de identificadores únicos. Actualización exclusiva de la documentación: Se han agrupado todas las funciones intrínsecas existentes y recién introducidas en las siguientes categorías según el tipo de tarea de procesamiento de datos que le ayudan a realizar: • Funciones intrínsecas para matrices • Funciones intrínsecas para la codificación y decodificación de datos • Función intrínseca para el cálculo del hash • Funciones intrínsecas para la manipulación de datos de JSON • Funciones intrínsecas para operaciones matemáticas • Función intrínseca para la operación de cadena • Función intrínseca para la generación de identificadores únicos • Función intrínseca para una operación genérica Para obtener más información, consulte Funciones intrínsecas.	31 de agosto de 20

Cambio	Descripción	Fecha de modificación
Actualización	Step Functions ha ampliado el soporte para las integraciones de AWS SDK al añadir tres AWS servicios más: AWS Billing ConductorAmazon GameSparks, yAmazon Pinpoint SMS and Voice V2. Para obtener más información, consulte Registro de cambios para las integraciones AWS de SDK compatibles.	26 de julio de 2022
Actualización exclusiva de la documentación	Se ha añadido un tema nuevo para incluir un resumen de todas las actualizaciones realizadas en las integraciones AWS del SDK compatibles con Step Functions. Para obtener más información, consulte Registro de cambios para las integraciones AWS de SDK compatibles	26 de julio de 2022
Actualización exclusiva de la documentación	AWS Step Functions La guía para desarrolladores ahora incluye detalles sobre las métricas de ejecución que se emiten específicamente para Express Workflows. Para obtener más información, consulte Métricas de ejecución para flujos de trabajo rápidos.	9 de junio de 2022

Cambio	Descripción	Fecha de modificación
Actualización	Mejoras en la consola de Step Functions La consola ahora incluye una página Detalles de la ejecución rediseñada con las siguientes mejoras: • Capacidad para identificar de un vistazo el motivo de una ejecución con error. • Dos nuevos modos de visualización para su máquina de estado: Vista de tabla y Vista de eventos. Estas vistas también le permiten aplicar filtros para ver solo la información que le interese. Además, puede ordenar el contenido de la Vista de eventos en función de las marcas temporales del evento. • Cambie entre las diferentes iteraciones de estado`Map` en el modo Vista de gráfico mediante una lista desplegab le o en la vista de árbol del modo Vista de tabla para los estados `Map`. • Vea información detallada sobre cada estado del flujo de trabajo, incluida la ruta completa de transferencia de datos de entrada y salida y los reintentos para los estados `Task` y `Parallel`. • Varias mejoras, incluida la opción de copiar el nombre del recurso de Amazon que se ejecuta en la máquina de estado, ver el recuento total de transiciones de la máquina de estado y exportar los detalles de la ejecución en formato JSON. Actualizaciones exclusivas de la documentación Se ha añadido un tema nuevo que explica los distintos tipos de información que se muestran en la página Detalles de la ejecución. Además, se ha añadido un tutorial para	9 de mayo de 2022

Cambio	Descripción	Fecha de modificación
	mostrar cómo examinar esta información. Para obtener más información, consulte: • Visualización y depuración de ejecuciones en la consola de Step Functions • Tutorial: Examinar las ejecuciones de máquinas de estados mediante la consola de Step Functions	
Actualización	Step Functions ahora ofrece una solución alternativa para evitar el problema de seguridad del suplente confuso, que surge cuando una entidad (un servicio o una cuenta) es obligada por una entidad diferente a realizar una acción. Para obtener más información, consulte: • Prevención del problema del suplente confuso entre servicios	2 de mayo de 2022

Cambio	Descripción	Fecha de modificación

Actualización	• Step Functions ha ampliado el soporte para las integraciones de AWS SDK al añadir 21 AWS servicios más. Para obtener más información, consulte: Integraciones de servicios AWS de SDK compatibles.
	• Actualizaciones exclusivas de la documentación:
	  • Se agregó una lista de todos los prefijos de excepción presentes en las excepciones que se generan al realizar por error una integración del servicio AWS SDK con Step Functions. Para obtener más información, consulte: Integraciones de servicios AWS de SDK compatibles.
	  • Se agregó una lista de todas las acciones de API no compatibles para las integraciones de SDK compatibles. AWS Para obtener más información, consulte: Acciones de API no admitidas para servicios admitidos .
	  • Se agregó una lista de todas las integraciones de AWS SDK compatibles que ahora están en desuso. Para obtener más información, consulte: Integraciones de servicios AWS SDK obsoletas.
		19 de abril de 2022

Nueva característica	Step Functions Local ahora admite la integración AWS del SDK y la simulación de integraciones de servicios. Para obtener más información, consulte: • Uso de integraciones de servicios simuladas	28 de enero de 2022

Cambio	Descripción	Fecha de modificación
Nueva característica	AWS Step Functions ahora admite la creación de una API REST de Amazon API Gateway con una máquina de estado express sincrónica como integración de backend mediante el. AWS Cloud Development Kit (AWS CDK) Para obtener más información, consulte: • Creación de una API REST de API Gateway con Synchronous Express State Machine mediante AWS CDK	10 de diciembre de 2021
Actualización	Step Functions ha agregado tres nuevos proyectos de muestra que ilustran la integración de Step Functions y la consola actualizada de Amazon Athena. Para obtener más información, consulte: • Ejecutar varias consultas (Amazon Athena, Amazon SNS) • Consulte conjuntos de datos de gran tamaño (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS) • Mantener los datos actualizados (Amazon Athena, Amazon S3,) AWS Glue	22 de noviembre de 2021
Nueva característica	Step Functions ha agregado compatibilidad con puntos de conexión de VPC de Amazon para flujos de trabajo rápidos sincrónicos. Para obtener más información, consulte: • Puntos de conexión de VPC para Step Functions	15 de noviembre de 2021

Cambio	Descripción	Fecha de modificación
Actualización	AWS Step Functions ha añadido tres nuevos proyectos de muestra que demuestran cómo utilizar la AWS Batch integración de Step Functions. Para obtener más información, consulte: • Haz un abanico de AWS Batch trabajo • AWS Batch con Lambda • Usar Step Functions y AWS Batch con control de errores	14 de octubre de 2021
Nueva característica	AWS Step Functions ha añadido integraciones con el AWS SDK, lo que permite utilizar las acciones de la API para todos los más de doscientos AWS servicios. Para obtener más información, consulte: • AWS Integraciones de servicios SDK • Recopile información sobre los buckets de Amazon S3 mediante integraciones de servicios de AWS SDK	30 de septiembre de 2021
Nueva característica	AWS Step Functions ha añadido un diseñador visual de flujos de trabajo, el AWS Step Functions Workflow Studio. Para obtener más información, consulte: • AWS Step Functions Estudio de flujo de trabajo • Aprenda a usar el AWS Step Functions Workflow Studio	17 de junio de 2021
Actualización	AWS Step Functions ha agregado cuatro nuevas API`StartBuildBatch` ,`StopBuildBatch` , `RetryBuildBatch` y`DeleteBuildBatch` , a la CodeBuild integración. Para obtener más información, consulte: • Llama AWS CodeBuild con Step Functions	4 de junio de 2021

Cambio	Descripción	Fecha de modificación
Nueva característica	AWS Step Functions ahora se integra con Amazon EventBridge. Para obtener más información, consulte: • Llama EventBridge con Step Functions • Políticas de IAM para Step Functions y Políticas de IAM para Amazon EventBridge • Un proyecto de muestra que ilustra cómo Envía un evento personalizado a EventBridge	14 de mayo de 2021
Actualización	AWS Step Functions ha añadido un nuevo proyecto de ejemplo que muestra cómo utilizar Step Functions y la API Amazon Redshift Data para ejecutar un flujo de trabajo de ETL/ELT. Para obtener más información, consulte: • Ejecutar flujos de trabajo de ETL/ELT con Amazon Redshift (Lambda, API de datos de Amazon Redshift)	16 de abril de 2021
Nueva característica	AWS Step Functions tiene un nuevo simulador de flujo de datos en la consola. Para obtener más información, consulte: • Consola de Step Functions	8 de abril de 2021
Nueva característica	AWS Step Functions ahora se integra con Amazon EMR en EKS. Para obtener más información, consulte: • Llame a Amazon EMR en EKS con AWS Step Functions	29 de marzo de 2021

Cambio	Descripción	Fecha de modificación
Actualización	Se ha agregado la compatibilidad con YAML para las definiciones de máquinas de estado a AWS Toolkit for Visual Studio Code y. AWS CloudFormation Para obtener más información, consulte: • Compatibilidad con formatos de definición • AWS Toolkit for Visual Studio Code	4 de marzo de 2021
Nueva característica	AWS Step Functions ahora se integra con AWS Glue DataBrew. Para obtener más información, consulte: • Gestione AWS Glue DataBrew trabajos con Step Functions • ¿Qué es AWS Glue DataBrew? en la guía para DataBrew desarrolladores.	6 de enero de 2021
Nueva característica	AWS Step Functions Los flujos de trabajo express sincrónicos ya están disponibles, lo que le proporciona una forma sencilla de organizar los microservicios. Para obtener más información, consulte: • Flujos de trabajo rápidos síncronos y asíncronos • Un proyecto de muestra que ilustra cómo Invocar flujos de trabajo rápidos sincrónicos • La documentación de la API StartSyncExecution.	24 de noviembre de 2020

Cambio	Descripción	Fecha de modificación
Nueva característica	AWS Step Functions ahora se integra con Amazon API Gateway. Para obtener más información, consulte: • Llamar a API Gateway con Step Functions • Políticas de IAM para Step Functions y Políticas de IAM para Amazon API Gateway • Un proyecto de muestra que ilustra cómo Hacer una llamada a API Gateway	17 de noviembre de 2020
Nueva característica	AWS Step Functions ahora se integra con Amazon Elastic Kubernetes Service. Para obtener más información, consulte: • Llamar a Amazon EKS con Step Functions • Políticas de IAM para Step Functions y Políticas de IAM para Amazon EKS • Un proyecto de muestra que ilustra cómo Administración de un clúster de Amazon EKS	16 de noviembre de 2020
Nueva característica	AWS Step Functions ahora se integra con Amazon Athena. Para obtener más información, consulte: • Llamar a Athena con Step Functions • Políticas de IAM para Step Functions y Políticas de IAM para Amazon Athena • Un proyecto de muestra que ilustra cómo Iniciar una consulta de Athena	22 de octubre de 2020

Cambio	Descripción	Fecha de modificación
Nueva característica	AWS Step Functions ahora es compatible con el seguimiento de end-to-end los flujos de trabajo AWS X-Ray, lo que le brinda una visibilidad total de las ejecuciones de las máquinas en estado y facilita el análisis y la depuración de sus aplicaciones distribuidas. Para obtener más información, consulte: • AWS X-Ray y Step Functions • Políticas de IAM para Step Functions y Políticas de IAM para AWS X-Ray • AWS Step Functions Referencia de la API • TracingConfiguration	14 de septiembre de 2020

Cambio	Descripción	Fecha de modificación

Actualización	AWS Step Functions ahora admite cargas útiles de hasta 256 KB de datos en forma de cadena codificada en UTF-8. Esto le permite procesar cargas más grandes en los flujos de trabajo estándar y rápidos.
	
	No es necesario cambiar las máquinas de estado existentes para poder utilizar las cargas más grandes. No obstante, tendrá que actualizar a las versiones más recientes del SDK de Step Functions y Local Runner para usar las API actualizadas. Para obtener más información, consulte:
	
	• Cuotas
	• the section called “Utilizar los ARN de Amazon S3 en lugar de pasar cargas de gran tamaño”
	• `States.DataLimitExceeded`
	• the section called “Cargas de CloudWatch Logs”
	• the section called “EventBridge cargas útiles”
	• AWS Step Functions Referencia de la API
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
	  • StateEnteredEventDetails
	  • StateExitedEventDetails
		3 de septiembre de 2020

Cambio	Descripción	Fecha de modificación
	• TaskSubmittedEventDetails • TaskSucceededEventDetails	

Cambio	Descripción	Fecha de modificación

Actualización	Se ha actualizado Amazon States Language de la siguiente manera:
	• Reglas de Choice ha añadido
	  • Un operador de comparación nulo, `IsNull`. `IsNull` comprueba el valor nulo de JSON y se puede utilizar para detectar si el resultado de un estado anterior es nulo o no.
	  • Se han agregado otros cuatro operadores nuevos, IsBoolean, IsNumeric y. IsString IsTimestamp
	  • Una prueba de la existencia o inexistencia de un campo mediante el operador `IsPresent` . `IsPresent` se puede usar para evitar errores `States.Runtime` cuando se intenta acceder a una clave que no existe.
	  • La coincidencia de patrones con comodines permite comparar cadenas con patrones con uno o varios comodines.
	  • Comparación entre dos variables para operadores de comparación compatibles.
	• Los valores de tiempo de espera y latido de un estado `Task` ahora se pueden proporcionar dinámicamente a partir de la entrada del estado en lugar de un valor fijo mediante los campos `TimeoutSecondsPath` y `HeartbeatSecondsPath` . Para obtener más información, consulte el estado Estado de la tarea.
	• El nuevo campo ResultSelector proporciona una forma de manipular el resultado de un estado antes de que se aplique `ResultPath` . El campo `ResultSelector` es un campo opcional en los estados Map, Parallel y Estado de la tarea.
		13 de agosto de 2020

Cambio	Descripción	Fecha de modificación
	• Se han añadido Funciones intrínsecas para permitir operaciones básicas sin estados `Task`. Se pueden usar funciones intrínsecas en los campos `Parameters` y `ResultSelector` .	
Actualización	AWS Step Functions ahora admite la llamada a la SageMaker `CreateProcessingJob` API de Amazon. Para obtener más información, consulte: • Administre SageMaker con Step Functions • Preprocesamiento de datos y entrenamiento de un modelo de machine learning, un proyecto de muestra que ilustra `CreateProcessingJob` .	4 de agosto de 2020
Nueva característica	AWS Step Functions ahora es compatible con AWS Serverless Application Model, lo que facilita la integración de la orquestación del flujo de trabajo en sus aplicaciones sin servidor. Para obtener más información, consulte: • AWS Step Functions y AWS SAM • AWS::Serverless::StateMachine • AWS SAM Plantillas de política	27 de mayo de 2020
Nueva característica	AWS Step Functions ha introducido una nueva invocación sincrónica para anidar las ejecuciones de Step Functions . La nueva invocación, `arn:aws:states:::states:startExecution.sync:2` , devuelve un objeto JSON. Se puede seguir utilizando la invocación original, `arn:aws:states:::states:startExecution.sync` , que devolverá una cadena de escape de JSON. Para obtener más información, consulte: • Gestione AWS Step Functions las ejecuciones como un servicio integrado	19 de mayo de 2020

Cambio	Descripción	Fecha de modificación
Nueva característica	AWS Step Functions ahora se integra con. AWS CodeBuild Para obtener más información, consulte: • Uso AWS Step Functions con otros servicios • Llama AWS CodeBuild con Step Functions • Integraciones optimizadas para Step Functions	5 de mayo de 2020
Nueva característica	Step Functions ahora es compatible en AWS Toolkit for Visual Studio Code, lo que facilita la creación y la visualización de flujos de trabajo basados en máquinas de estados sin salir del editor de código.	31 de marzo de 2020
Actualización	Ahora puede configurar el registro en Amazon CloudWatch Logs para flujos de trabajo estándar. Para obtener más información, consulte: • Registro mediante CloudWatch Logs	25 de febrero de 2020
Nueva característica	AWS Step Functions ahora se puede acceder a ellos sin necesidad de una dirección IP pública, directamente desde Amazon Virtual Private Cloud (VPC). Para obtener más información, consulte: • Puntos de conexión de VPC para Step Functions	23 de diciembre de 2019

Cambio	Descripción	Fecha de modificación

Nueva característica	Los flujos de trabajo rápidos son un nuevo tipo de flujo de trabajo, adecuado para cargas de trabajo de procesamiento de eventos de un volumen elevado como la incorporación de datos de IoT, el streaming de la transformación y el procesamiento de los datos y los backends de aplicaciones móviles.
	
	Para obtener más información, consulte los nuevos temas actualizados.
	
	• Flujos de trabajo estándar en comparación con flujos de trabajo rápidos
	  • Garantías de ejecución
	• Uso AWS Step Functions con otros servicios
	  • Integraciones optimizadas para Step Functions
	• Procesar mensajes de un volumen elevado desde Amazon SQS (flujos de trabajo rápidos)
	• Ejemplo de punto de comprobación selectivo (flujos de trabajo rápidos)
	• Cuotas
	  • Cuotas
	• Registro mediante CloudWatch Logs
	• AWS Step Functions Referencia de la API
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
		3 de diciembre de 2019

Cambio	Descripción	Fecha de modificación
	• ListExecutions • ListStateMachines • StartExecution • CloudWatchLogsLogGroup • LogDestination • LoggingConfiguration	
Nueva característica	AWS Step Functions ahora se integra con Amazon EMR. Para obtener más información, consulte: • Uso AWS Step Functions con otros servicios • Llamar a Amazon EMR con Step Functions • Integraciones optimizadas para Step Functions	19 de noviembre de 2019

Actualización	AWS Step Functions ha lanzado el SDK de ciencia de datos de AWS Step Functions. Para obtener más información, consulte lo siguiente.
	• Proyecto en Github
	• Documentación de SDK
	• Los siguientes ejemplos de cuadernos, que están disponibles en la SageMakerconsola y en el GitHub proyecto relacionado.
	  • `hello_world_workflow.ipynb`
	  • `machine_learning_workflow_abalone.ipynb`
	  • `training_pipeline_pytorch_mnist.ipynb`
		7 de noviembre de 2019

Cambio	Descripción	Fecha de modificación
Actualización	Step Functions ahora admite más acciones de API para Amazon SageMaker e incluye dos nuevos proyectos de muestra para demostrar la funcionalidad. Para obtener más información, consulte lo siguiente. • Administre SageMaker con Step Functions • Uso AWS Step Functions con otros servicios • Entrenamiento de un modelo de aprendizaje automático • Ajuste de un modelo de aprendizaje automático	3 de octubre de 2019
Nueva característica	Step Functions permite iniciar nuevas ejecuciones de flujo de trabajo llamando a `StartExecution` como API de servicio integrada. Consulte: • Iniciar ejecuciones de flujo de trabajo desde un estado de tarea • Gestione AWS Step Functions las ejecuciones como un servicio integrado • Uso AWS Step Functions con otros servicios • Políticas de IAM para iniciar ejecuciones de flujo de trabajo de Step Functions	12 de agosto de 2019

Cambio	Descripción	Fecha de modificación
Nueva característica	Step Functions incluye la capacidad de transferir un token de tarea para los servicios integrados y pausar la ejecución token de tarea hasta que se devuelve con `SendTaskSuccess` o `SendTaskFailure` . Consulte: • Patrones de integración de servicios • Cómo esperar una devolución de llamada con el token de tarea • Ejemplo de patrón de devolución de llamadas (Amazon SQS, Amazon SNS, Lambda) • Integraciones optimizadas para Step Functions • Implementación de un proyecto de aprobación humana de ejemplo • Métricas de integración de servicios Step Functions ahora proporciona una manera de acceder a información dinámica acerca de su ejecución actual directamente en el campo `"Parameters"` de una definición de estado. Consulte: • Objeto Context (Contexto) • Cómo transferir nodos del objeto context como parámetros	23 de mayo de 2019
Nueva característica	Step Functions admite CloudWatch eventos para cambios de estado de ejecución, consulte: • EventBridge (CloudWatch Eventos) para cambios en el estado de ejecución de Step Functions • Guía del usuario de Amazon CloudWatch Events	8 de mayo de 2019

Cambio	Descripción	Fecha de modificación
Nueva característica	Step Functions admite los permisos de IAM que se basan en etiquetas. Para obtener más información, consulte: • Etiquetado en Step Functions • Políticas basadas en etiquetas	5 de marzo de 2019
Nueva característica	Step Functions Local ya está disponible. Puede ejecutar Step Functions en su equipo local para las pruebas y el desarrollo. Step Functions Local está disponible para su descarga como una aplicación Java o como una imagen de Docker. Consulte Probar máquinas de estado de forma local.	4 de febrero de 2019
Nueva característica	AWS Step Functions ya está disponible en las regiones de Beijing y Ningxia. Consulte Regiones de admitidas.	15 de enero de 2018
Nueva característica	Step Functions es compatible con el etiquetado de recursos para ayudarle a hacer un seguimiento de la asignación de costos. Puede etiquetar máquinas de estado en la página Detalles o mediante acciones de la API. Consulte Etiquetado en Step Functions.	7 de enero de 2019
Nueva característica	AWS Step Functions ya está disponible en las regiones de Europa (París) y Sudamérica (São Paulo). Consulte Regiones de admitidas.	13 de diciembre de 2018
Nueva característica	AWS Step Functions ya está disponible en la región Europa (Estocolmo). Consulte Regiones de admitidas para obtener una lista de las regiones admitidas.	12 de diciembre de 2018

Cambio	Descripción	Fecha de modificación
Nueva característica	Step Functions ahora se integra con algunos AWS servicios. Ahora puede llamar directamente y pasar parámetros a la API de estos servicios integrados desde un estado de tarea en Amazon States Language. Para obtener más información, consulte: • Uso AWS Step Functions con otros servicios • Cómo pasar parámetros a una API de servicio • Integraciones optimizadas para Step Functions	29 de noviembre de 2018
Actualización	Se ha mejorado la descripción de `TimeoutSeconds` y de `HeartbeatSeconds` en la documentación de los estados de tareas. Consulte Estado de la tarea.	24 de octubre de 2018
Actualización	Se ha mejorado la descripción del límite del tamaño máximo del historial de ejecuciones y se ha proporcionado un enlace para el tema de prácticas recomendadas. • Cuotas relacionadas con ejecuciones de máquinas de estado • Evitar alcanzar la cuota de historial	17 de octubre de 2018
Actualización	Se agregó un nuevo tutorial a la AWS Step Functions documentación: consulteIniciar ejecución de una máquina de estado en respuesta a eventos de Amazon S3.	25 de septiembre de 2018
Actualización	Se ha eliminado la entrada Máximo de ejecuciones mostradas en la consola de funciones de paso de la documentación de límites. Consulte Cuotas.	13 de septiembre de 2018
Actualización	Se agregó un tema de mejores prácticas a la AWS Step Functions documentación sobre cómo mejorar la latencia al sondear las tareas de actividad. Consulte Evitar la latencia al sondear tareas de actividad.	30 de agosto de 2018

Cambio	Descripción	Fecha de modificación
Actualización	Se ha mejorado el AWS Step Functions tema sobre las actividades y los trabajadores activos. Consulte Actividades.	29 de agosto de 2018
Actualización	Se ha mejorado el AWS Step Functions tema de CloudTrail la integración. Consulte Grabación de llamadas a la API con AWS CloudTrail.	7 de agosto de 2018
Actualización	Se agregaron ejemplos de JSON al AWS CloudFormation tutorial. Consulte Creación de una máquina de estado Lambda para Step Functions mediante AWS CloudFormation.	23 de junio de 2018
Actualización	Se ha añadido un nuevo tema sobre control de errores de servicio en Lambda. Consulte Gestionar excepciones de servicio de Lambda.	20 de junio de 2018
Nueva característica	AWS Step Functions ya está disponible en la región Asia-Pacífico (Bombay). Consulte Regiones de admitidas para obtener una lista de las regiones admitidas.	28 de junio de 2018
Nueva característica	AWS Step Functions ya está disponible en la región AWS GovCloud (EE. UU.-Oeste). Consulte Regiones de admitidas para obtener una lista de las regiones admitidas. Para obtener información sobre el uso de Step Functions en la región AWS GovCloud (EE. UU.-Oeste), consulte AWS GovCloud (US).	28 de junio de 2018
Actualización	Se ha mejorado la documentación sobre control de errores para estados `Parallel`. Consulte Control de errores.	20 de junio de 2018

Cambio	Descripción	Fecha de modificación
Actualización	Se ha mejorado la documentación sobre procesamiento de entrada y salida en Step Functions. Aprenda a utilizar `InputPath` , `ResultPath` y `OutputPath` para controlar el flujo de JSON a través de sus flujos de trabajo, estados y tareas. Consulte: • Procesamiento de entrada y salida en Step Functions • ResultPath	7 de junio de 2018
Actualización	Ejemplos de código mejorado para estados Parallel. Consulte Parallel.	4 de junio de 2018
Nueva característica	Ahora puede supervisar las métricas de las API y los servicios en CloudWatch. Consulte Supervisión de las funciones de Step Functions mediante CloudWatch.	25 de mayo de 2018
Actualización	`StartExecution` , `StopExecution` y `StateTransition` ahora han aumentado los límites de limitación controlada en las siguientes regiones: • Este de EE. UU. (Norte de Virginia) • Oeste de EE. UU. (Oregón) • Europa (Irlanda) Para más información, consulte Cuotas.	16 de mayo de 2018
Nueva característica	AWS Step Functions ya está disponible en las regiones EE.UU. Oeste (Norte de California) y Asia Pacífico (Seúl). Consulte Regiones de admitidas para obtener una lista de las regiones admitidas.	5 de mayo de 2018

Cambio	Descripción	Fecha de modificación
Actualización	Actualización de los procedimientos e imágenes para que se correspondan con los cambios de la interfaz.	25 de abril de 2018
Actualización	Se agregó un nuevo tutorial que muestra cómo iniciar una nueva ejecución para continuar el trabajo. Consulte Continuar las ejecuciones de flujos de trabajo de ejecución prolongada como una nueva ejecución. En este tutorial se describe un patrón de diseño que puede ayudarle a evitar algunas limitaciones de servicio. Consulte Evitar alcanzar la cuota de historial.	19 de abril de 2018
Actualización	Se ha mejorado la introducción a la documentación de estados añadiendo información conceptual sobre máquinas de estado. Consulte Estados.	9 de marzo de 2018
Actualización	Además de HTML, PDF y Kindle, la Guía para AWS Step Functions desarrolladores está disponible en GitHub. Para dejar un comentario, selecciona el GitHub icono situado en la esquina superior derecha. 	2 de marzo de 2018
Actualización	Se ha añadido un tema que describe otros recursos relacionados con Step Functions. Consulte Información relacionada.	20 de febrero de 2018

Cambio	Descripción	Fecha de modificación

Nueva característica	• Al crear una nueva máquina de estado, debe confirmar que AWS Step Functions creará un rol de IAM que permita el acceso a las funciones de Lambda.
	• Se han actualizado los siguientes tutoriales para reflejar pequeños cambios en el flujo de trabajo de creación de máquinas de estado:
	  • Creación de una máquina de estado de Step Functions que utilice Lambda
	  • Crear una máquina de estado de actividad con Step Functions
	  • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos
	  • Itera un bucle con Lambda
		19 de febrero de 2018

Actualización	Se ha añadido un tema que describe un ejemplo de proceso de trabajo de actividad escrito en Ruby. Esta implementación se puede utilizar para crear un proceso de trabajo de actividad de Ruby directamente, o como un patrón de diseño para la creación de un proceso de trabajo de actividad en otro lenguaje. Consulte Ejemplo de proceso de trabajo de actividad en Ruby.	6 de febrero de 2018
Actualización	Se ha añadido un nuevo tutorial que describe un patrón de diseño que utiliza una función de Lambda para iterar un contador. Consulte Creación de una máquina de estado de Step Functions que utilice Lambda.	31 de enero de 2018

Cambio	Descripción	Fecha de modificación
Actualización	Se ha actualizado el contenido relacionado con los permisos de IAM para incluir las API `DescribeStateMachineForExecution` y `UpdateStateMachine`. Consulte Creación de permisos granulares de IAM para usuarios no administradores.	26 de enero de 2018
Actualización	Se han añadido nuevas regiones disponibles: Canadá (centro) y Asia Pacífico (Singapur). Consulte Regiones de admitidas.	25 de enero de 2018
Actualización	Se han actualizado los tutoriales y procedimientos para reflejar que IAM permite seleccionar Step Functions como rol.	24 de enero de 2018
Actualización	Se ha añadido el tema nuevo Prácticas recomendadas que sugiere que no se pasen cargas de gran tamaño entre los estados. Consulte Utilizar los ARN de Amazon S3 en lugar de pasar cargas de gran tamaño.	23 de enero de 2018
Actualización	Se han corregido los siguientes procedimientos para adaptarlos a la interfaz actualizada de creación de una máquina de estado: • Creación de una máquina de estado de Step Functions que utilice Lambda • Crear una máquina de estado de actividad con Step Functions • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos	17 de enero de 2018

Cambio	Descripción	Fecha de modificación
Nueva característica	Es posible utilizar proyectos de ejemplo para aprovisionar rápidamente máquinas de estado y todos los recursos de AWS relacionados. Consulte Proyectos de muestra para Step Functions. Los proyectos de muestra disponibles son los siguientes: • Encuesta sobre el estado del trabajo (Lambda,) AWS Batch • Temporizador de tareas (Lambda, Amazon SNS) **Note** Estos proyectos de muestra y la documentación relacionada sustituyen a los tutoriales en los que se describe la implementación de la misma funcionalidad.	11 de enero de 2018
Actualización	Se ha añadido una sección Prácticas recomendadas que incluye información sobre cómo evitar que las ejecuciones se bloqueen. Consulte Prácticas recomendadas para Step Functions.	5 de enero de 2018
Actualización	Se ha añadido una nota sobre cómo pueden afectar los reintentos al precio: **Note** Los reintentos se tratan como transiciones de estado. Para obtener información sobre cómo las transiciones de estado afectan a la facturación, consulte Precios de Step Functions	8 de diciembre de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha añadido información relacionada con los nombres de recursos: **Note** Step Functions le permite crear nombres para máquinas de estados, ejecuciones y actividades, así como etiquetas que contienen caracteres no ASCII. Estos nombres que no son ASCII no funcionan con Amazon. CloudWatch Para asegurarse de que puede realizar un seguimiento de CloudWatch las métricas, elija un nombre que utilice únicamente caracteres ASCII.	6 de diciembre de 2017
Actualización	Se ha mejorado la información general sobre seguridad y se ha añadido un tema sobre permisos granulares de IAM. Consulte Seguridad en AWS Step Functions y Creación de permisos granulares de IAM para usuarios no administradores.	27 de noviembre de 2017
Nueva característica	Puede actualizar una máquina de estado existente. Consulte Actualizar la máquina de estados.	15 de noviembre de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha añadido una nota para aclarar errores `Lambda.Unknown` y se ha vinculado a la documentación de Lambda en las secciones siguientes: • Nombres de error • Paso 3: Crear una máquina de estado con un campo Catch **Note** Los errores no controlados en Lambda se notifican como `Lambda.Unknown` en el resultado del error. Entre ellos se incluyen out-of-memory los errores y los tiempos de espera de las funciones. Puede buscar coincidencias de estos errores con `Lambda.Unknown` , `States.ALL` o `States.TaskFailed` para controlarlos. Cuando Lambda alcanza el número máximo de invocaciones, el error es. `Lambda.TooManyRequestsException` Para obtener más información acerca de errores de función de Lambda, consulte Tratamiento de errores y reintentos automáticos en la Guía para desarrolladores de AWS Lambda .	17 de octubre de 2017
Actualización	Se han corregido y aclarado las instrucciones de IAM y se han actualizado todas las capturas de pantalla en todos los tutoriales.	11 de octubre de 2017

Cambio	Descripción	Fecha de modificación

Actualización	• Se han añadido nuevas capturas de pantalla para que los resultados de ejecución de máquina de estado reflejen cambios en la consola de Step Functions. Se han reescrito las instrucciones de Lambda en los siguientes tutoriales para reflejar los cambios en la consola de Lambda:
	  • Creación de una máquina de estado de Step Functions que utilice Lambda
	  • Creación de un sondeador del estado de un trabajo
	  • Creación de un temporizador de tareas
	  • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos
	• Se ha corregido y aclarado la información sobre la creación de máquinas de estado en las secciones siguientes:
	  • Crear una máquina de estado de actividad con Step Functions
		6 de octubre de 2017

Actualización	Se han reescrito las instrucciones de IAM en las siguientes secciones para reflejar los cambios en la consola de IAM: • Creación de un rol de IAM para la máquina de estado • Creación de una máquina de estado de Step Functions que utilice Lambda • Creación de un sondeador del estado de un trabajo • Creación de un temporizador de tareas • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos • Crear una API de Step Functions utilizando API Gateway	5 de octubre de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha reescrito la sección Datos de la máquina de estado.	28 de septiembre de 2017
Nueva característica	Los límites de limitación controlada de las acciones de la API se han aumentado para todas las regiones donde Step Functions está disponible.	18 de septiembre de 2017
Actualización	• Se ha corregido y clarificado la información sobre cómo iniciar nuevas ejecuciones en todos los tutoriales. • Se ha corregido y clarificado la información de la sección Cuotas relacionadas con las cuentas.	14 de septiembre de 2017
Actualización	Se han reescrito los siguientes tutoriales para reflejar los cambios en la consola de Lambda: • Creación de una máquina de estado de Step Functions que utilice Lambda • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos • Creación de un sondeador del estado de un trabajo	28 de agosto de 2017
Nueva característica	Step Functions está disponible en Europa (Londres).	23 de agosto de 2017
Nueva característica	Los flujos de trabajo visuales de las máquinas de estado le permiten ampliar, reducir y centrar el gráfico.	21 de agosto de 2017

Cambio	Descripción	Fecha de modificación
Nueva característica	**Important** Una ejecución no puede utilizar el nombre de otra ejecución durante 90 días. Al realizar varias `StartExecution` llamadas con el mismo nombre, la nueva ejecución no se ejecuta. Para obtener más información, consulte el parámetro de solicitud `name` de la acción de la API de `StartExecution` en la Referencia de la API de AWS Step Functions.	18 de agosto de 2017
Actualización	Se ha añadido información acerca de un método alternativo para pasar el ARN de la máquina de estado en el tutorial Crear una API de Step Functions utilizando API Gateway.	17 de agosto de 2017
Actualización	Se ha añadido el nuevo tutorial Creación de un sondeador del estado de un trabajo.	10 de agosto de 2017
Nueva característica	• Step Functions emite la `ExecutionThrottled` CloudWatch métrica. Para obtener más información, consulte Supervisión de las funciones de Step Functions mediante CloudWatch. • Se ha agregado la sección Cuotas relacionadas con la limitación controlada de estados.	3 de agosto de 2017
Actualización	Se han actualizado las instrucciones de la sección Paso 2: Crear un rol de IAM para API Gateway.	18 de julio de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha corregido y clarificado la información de la sección Choice.	23 de junio de 2017
Actualización	Se agregó información sobre el uso de recursos en otras AWS cuentas a los siguientes tutoriales: • Creación de una máquina de estado de Step Functions que utilice Lambda • Creación de una máquina de estado Lambda para Step Functions mediante AWS CloudFormation • Crear una máquina de estado de actividad con Step Functions • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos	22 de junio de 2017
Actualización	Se ha corregido y clarificado la información de las siguientes secciones: • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos • Estados • Control de errores en Step Functions	21 de junio de 2017
Actualización	Se han reescrito todos los tutoriales para adaptarlos a la actualización de la consola de Step Functions.	12 de junio de 2017
Nueva característica	Step Functions está disponible en la región de Asia Pacífico (Sídney).	8 de junio de 2017
Actualización	Se ha reestructurado la sección Lenguaje de estados de Amazon.	7 de junio de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha corregido y clarificado la información de la sección Crear una máquina de estado de actividad con Step Functions.	6 de junio de 2017
Actualización	Se han corregido los ejemplos de código de la sección Ejemplos de máquina de estado que usan Retry y Catch.	5 de junio de 2017
Actualización	Reestructuró esta guía utilizando estándares de AWS documentación.	31 de mayo de 2017
Actualización	Se ha corregido y clarificado la información de la sección Parallel.	25 de mayo de 2017
Actualización	Se han combinado las secciones Rutas y Filtros en la sección Procesamiento de entrada y salida en Step Functions.	24 de mayo de 2017
Actualización	Se ha corregido y clarificado la información de la sección Supervisión de las funciones de Step Functions mediante CloudWatch.	15 de mayo de 2017
Actualización	Se ha actualizado el código del proceso de trabajo `GreeterActivities.java` en el tutorial Crear una máquina de estado de actividad con Step Functions.	9 de mayo de 2017
Actualización	Se ha añadido un vídeo introductorio en la sección ¿Qué es AWS Step Functions?.	19 de abril de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha corregido y clarificado la información de los siguientes tutoriales: • Creación de una máquina de estado de Step Functions que utilice Lambda • Crear una máquina de estado de actividad con Step Functions • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos	19 de abril de 2017
Actualización	Se ha añadido información sobre las plantillas de Lambda en los tutoriales Creación de una máquina de estado de Step Functions que utilice Lambda y Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos.	6 de abril de 2017
Actualización	Se ha cambiado el límite "Maximum input or result data size" a "Maximum input or result data size for a task, state, or execution" (32 768 caracteres). Para obtener más información, consulte Cuotas relacionadas con ejecuciones de tarea.	31 de marzo de 2017
Nueva característica	• Step Functions permite ejecutar máquinas de estados al establecer Step Functions como objetivos de Amazon CloudWatch Events.	21 de marzo de 2017
Nueva característica	• Step Functions permite el control de errores de funciones de Lambda como método de control de errores preferido. • Se ha actualizado el tutorial Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos y la sección Control de errores en Step Functions.	16 de marzo de 2017

Cambio	Descripción	Fecha de modificación
Nueva característica	Step Functions está disponible en Europa (Fráncfort).	7 de marzo de 2017
Actualización	Se han reorganizado los temas de la tabla de contenido y se han actualizado los siguientes tutoriales: • Creación de una máquina de estado de Step Functions que utilice Lambda • Crear una máquina de estado de actividad con Step Functions • Tratamiento de condiciones de error mediante una máquina de estado de funciones por pasos	23 de febrero de 2017
Nueva característica	• La página Máquinas de estado de la consola de Step Functions incluye los botones Copiar en nuevo y Eliminar. • Se han actualizado las capturas de pantalla para adaptarlas a los cambios de la consola.	23 de febrero de 2017
Nueva característica	• Step Functions admite la creación de API mediante API Gateway. • Se ha añadido el tutorial Crear una API de Step Functions utilizando API Gateway.	14 de febrero de 2017
Nueva característica	• Step Functions admite la integración con AWS CloudFormation. • Se ha añadido el tutorial Creación de una máquina de estado Lambda para Step Functions mediante AWS CloudFormation.	10 de febrero de 2017

Cambio	Descripción	Fecha de modificación
Actualización	Se ha clarificado el comportamiento actual de los campos `ResultPath` y `OutputPath` en relación con los estados `Parallel`.	6 de febrero de 2017
Actualización	• Se han clarificado las restricciones de nomenclatura de máquinas de estado en los tutoriales. • Se han corregido algunos ejemplos de código.	5 de enero de 2017
Actualización	Se han actualizado los ejemplos de las funciones de Lambda de acuerdo con el último modelo de programación.	9 de diciembre de 2016
Versión inicial	Versión inicial de AWS Step Functions.	1 de diciembre de 2016

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.